



Benutzer-Leitfaden

Amazon EC2 Auto Scaling



Amazon EC2 Auto Scaling: Benutzer-Leitfaden

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon EC2 Auto Scaling?	1
Funktionen von Amazon EC2 Auto Scaling	2
Preise für Amazon EC2 Auto Scaling	4
Erste Schritte	4
Arbeiten mit Auto-Scaling-Gruppen	4
Vorteile von Auto Scaling	5
Beispiel: Abdecken des Variablenbedarfs	6
Beispiel: Architektur für eine Web-App	8
Beispiel: Aufteilen von Instances in mehrere Availability Zones	9
Instance-Lebenszyklus	13
Horizontale Skalierung	14
In Betrieb genommene Instances	15
Scale-In	15
Trennen einer Instance	17
Hinzufügen einer Instance	17
Lebenszyklus-Hooks	17
Aktivieren und Deaktivieren des Standby-Status	18
Amazon EC2 Auto Scaling Scaling-Kontingente	18
Drosselung von Anfragen für die Amazon EC2 Auto Scaling Scaling-API	20
EC2-Beendigungsraten	20
Sonstige -Services	20
Einrichten	22
Bereiten Sie sich auf die Nutzung des vor AWS CLI	22
Erste Schritte	23
Tutorial: Erstellen Sie Ihre erste Auto Scaling Scaling-Gruppe	24
Vorbereitung auf den Walkthrough	24
Schritt 1: Eine Startvorlage erstellen	25
Schritt 2: Eine Auto-Scaling-Gruppe mit einer einzelnen Instance erstellen	26
Schritt 3: Überprüfen Ihrer Auto-Scaling-Gruppe	27
Schritt 4: Beenden einer Instance in Ihrer Auto-Scaling-Gruppe	28
Schritt 5: Nächste Schritte	29
Schritt 6: Bereinigen	30
Tutorial: Einrichten einer skalierten Anwendung mit Load Balancing	31
Voraussetzungen	33

Schritt 1: Einrichten einer Startvorlage oder Startkonfiguration	34
Schritt 2: Erstellen einer Auto-Scaling-Gruppe	38
Schritt 3: Überprüfen Sie, ob Ihr Load Balancer angefügt ist	39
Schritt 4: Nächste Schritte	40
Schritt 5: Bereinigen	41
Zugehörige Ressourcen	42
Startvorlagen	43
Berechtigungen für die Arbeit mit Startvorlagen	44
Von Startvorlagen unterstützte API-Operationen	44
Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe	45
So erstellen Sie eine Startvorlage (Konsole)	45
So ändern Sie die Standardeinstellungen für die Netzwerkschnittstelle (Konsole)	49
Ändern Sie die Speicherkonfiguration (Konsole)	51
Erstellen Sie eine Startvorlage anhand einer vorhandenen Instance (Konsole)	55
Zugehörige Ressourcen	55
Einschränkungen	56
Erstellen einer Startvorlage mithilfe erweiterter Einstellungen	56
Erforderliche Einstellungen	56
Erweiterte Einstellungen	57
Request Spot Instances	62
Capacity Blocks für ML	64
Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten	69
Schritt 1: Suchen Sie Auto-Scaling-Gruppen, die Startkonfigurationen verwenden	70
Schritt 2: Kopieren einer Startkonfiguration in eine Startvorlage	72
Schritt 3: Aktualisieren einer Auto-Scaling-Gruppe zum Verwenden einer Startvorlage	73
Schritt 4: Ersetzen Ihrer Instances	74
Zusätzliche Informationen	75
Migrieren Sie CloudFormation Stacks, um Vorlagen zu starten	75
Auto-Scaling-Gruppen finden, die eine Startkonfiguration verwenden	76
Aktualisieren eines Stacks zur Verwendung einer Startvorlage	76
Das Aktualisierungsverhalten von Stack-Ressourcen verstehen	81
Verfolgen Sie die Migration	81
Referenz für die Abbildung der Startkonfiguration	82
AWS CLI Beispiele für die Arbeit mit Startvorlagen	83
Beispielverwendung	84
Erstellen einer grundlegenden Startvorlage	85

Angeben von Tags, die Instances beim Start kennzeichnen	86
Angeben einer IAM-Rolle, die an Instances übergeben wird	86
Zuweisen einer öffentlichen IP-Adresse	86
Angeben eines Benutzerdatenskripts, das Instances beim Start konfiguriert	87
Angeben einer Blockgerät-Zuweisung für ein AMI	87
Festlegen von Dedicated Hosts zur Bereitstellung von Softwarelizenzen externer Anbieter	88
Angeben einer vorhandenen Netzwerkschnittstelle	88
Erstellen mehrerer Netzwerkschnittstellen	88
Verwalten Ihrer Startvorlagen	89
Aktualisieren einer Auto-Scaling-Gruppe zum Verwenden einer Startvorlage	92
Verwenden Sie Systems Manager Manager-Parameter anstelle von AMI IDs	93
Erstellen Sie eine Startvorlage, die einen Parameter für das AMI angibt	94
Stellen Sie sicher, dass eine Startvorlage die richtige AMI-ID erhält	99
Zugehörige Ressourcen	99
Einschränkungen	100
Startkonfigurationen	101
Erstellen einer Startkonfiguration	102
Erstellen einer Startkonfiguration	103
Konfigurieren von IMDS	106
Erstellen einer Startkonfiguration aus einer EC2-Instance	108
Ändern einer Startkonfiguration	113
Auto-Scaling-Gruppen	115
Erstellen Sie Auto-Scaling-Gruppen mit Startvorlagen	117
Erstellen einer Gruppe mithilfe einer Startvorlage	117
Erstellen einer Gruppe mithilfe des EC2-Startassistenten	121
Mehrere Instance-Typen und Kaufoptionen verwenden	126
Erstellen Sie Auto-Scaling-Gruppen mit Startkonfigurationen	177
Eine Gruppe mithilfe einer Startkonfiguration erstellen	178
Erstellen Sie eine Gruppe aus einer Instanz mit AWS CLI	181
Synchrone Bereitstellung	187
Synchrone Bereitstellung und asynchrone Skalierung	188
Einschränkungen und Überlegungen	189
Instances mit synchroner Bereitstellung werden gestartet	190
Aktualisieren einer Auto-Scaling-Gruppe	193
Aktualisieren von Auto-Scaling-Instances	195
Auto Scaling Scaling-Gruppenzuweisungsstrategie und Kapazitätsänderungen	196

Markieren von Gruppen und Instances	197
Einschränkungen für die Tag-Benennung und -Nutzung	198
Tagging-Lebenszyklus von EC2-Instances	198
Markieren Ihrer Auto-Scaling-Gruppen	199
Löschen von Tags	202
Tags für Sicherheit	203
Steuern des Zugriffs auf Tags	204
Verwenden Sie Tags, um Auto-Scaling-Gruppen zu filtern	205
Wartungsrichtlinien für Instances	209
-Übersicht	210
Festlegen einer Instance-Wartungsrichtlinie für Ihre Gruppe	218
Lebenszyklus-Hooks	223
Verfügbarkeit von Lebenszyklus-Hooks	224
Überlegungen und Einschränkungen	224
Zugehörige Ressourcen	227
So funktionieren Lifecycle-Hooks in Auto Scaling Scaling-Gruppen	227
Vorbereiten des Hinzufügens eines Lebenszyklus-Hook	231
Steuern Sie die Aufbewahrung von Instances mit Richtlinien für den Instance- Lebenszyklus	241
Abrufen des Ziellebenszyklus-Status	247
Fügen Sie Lifecycle-Hooks zu Ihrer Auto Scaling Scaling-Gruppe hinzu	249
Eine Lebenszyklusaktion in einer Auto Scaling Scaling-Gruppe abschließen	253
Tutorial: Verwenden Sie Instanz-Metadaten, um den Lebenszyklusstatus abzurufen	255
Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft	265
Warm-Pools	275
Schlüsselkonzepte	276
Voraussetzungen	279
Aktualisieren der Instances in einem warmen Pool	280
Zugehörige Ressourcen	281
Einschränkungen	281
Verwenden von Lebenszyklus-Hooks	282
Erstellen eines warmen Pools für eine Auto-Scaling-Gruppe	287
Anzeigen des Status der Zustandsprüfung	289
AWS CLI Beispiele für die Arbeit mit warmen Pools	293
Auto Scaling Scaling-Gruppenzonenverschiebung	296
Auto Scaling Scaling-Konzepte für Gruppen mit zonaler Verschiebung	296

So funktioniert Zonal Shift für Auto Scaling Scaling-Gruppen	297
Bewährte Methoden für die Verwendung von Zonal Shift	299
Aktivieren Sie Zonal Shift mit dem oder AWS-Managementkonsole AWS CLI	300
Verteilung von Availability Zones	303
Instanzen trennen und anhängen	304
Überlegungen zum Trennen von Instanzen	305
Überlegungen zum Anhängen von Instances	305
Verschieben Sie eine Instance mithilfe von Trennen und Anhängen in eine andere Gruppe .	306
Vorübergehendes Entfernen von Instances	311
So funktioniert der Standby-Status	312
Überlegungen	313
Zustand einer Instance im Standby-Status	314
Entfernen Sie eine Instance vorübergehend, indem Sie sie in den Standby-Modus versetzen	312
Löschen der Auto-Scaling-Infrastruktur	319
Löschen Ihrer Auto-Scaling-Gruppe	319
(Optional) Löschen der Startkonfiguration	320
(Optional) Löschen Sie die Startvorlage	321
(Optional) Löschen des Load Balancers und der Zielgruppen	321
(Optional) Alarme löschen CloudWatch	322
Schutz vor dem Löschen von Ressourcen	323
Ersetzen Sie Ihre Instanzen	330
Instance-Aktualisierung	331
Wie funktioniert eine Instanzaktualisierung	332
Die Standardwerte verstehen	339
Starten einer Instance-Aktualisierung	342
Überwachen Sie eine Instanzaktualisierung	356
Ersetzen Sie Root-Volumes	359
Abbrechen einer Instance-Aktualisierung	364
Änderungen mit einem Rollback rückgängig machen	365
Verwenden der Funktion zum Überspringen des Abgleichs	371
Hinzufügen von Checkpoints	381
Maximale Lebensdauer von Instances	387
Überlegungen	387
Maximale Lebensdauer von Instances festlegen	388
Einschränkungen	389

Skalieren Ihrer Gruppe	391
Wählen Sie Ihre Skalierungsmethode aus	392
Festlegen von Skalierungslimits	393
Standardmäßige Instance-Vorbereitungszeit einstellen	395
Leistungsaspekte der Skalierung	396
Wählen Sie die Standardzeit für das Aufwärmen der Instanz	397
Aktivieren Sie das Standard-Instance-Warmup für eine Gruppe	398
Überprüfen Sie die standardmäßige Aufwärmzeit der Instanz für eine Gruppe	400
Suchen Sie nach Skalierungsrichtlinien mit einer zuvor festgelegten Aufwärmzeit für Instanzen	401
Löschen Sie die zuvor festgelegte Instance-Vorbereitung für eine Skalierungsrichtlinie	402
Manuelle Skalierung	403
Ändern der gewünschten Kapazität einer Auto-Scaling-Gruppe	403
Beenden einer Instance in Ihrer Auto-Scaling-Gruppe (AWS CLI)	407
Geplante Skalierung	408
So funktioniert die geplante Skalierung	409
Wiederkehrende Zeitpläne	410
Zeitzone	410
Überlegungen	411
Einschränkungen	412
Eine geplante Aktion erstellen	412
Details zu geplanten Aktionen anzeigen	414
Löschen einer geplanten Aktion	415
Dynamische Skalierung	416
Funktionsweise von dynamischen Skalierungsrichtlinien	417
Mehrere dynamische Skalierungsrichtlinien	419
Skalierungsrichtlinien für die Ziel-Nachverfolgung	420
Schrittweise und einfache Skalierungsrichtlinien	439
Ruhephasen für die Skalierung	457
Skalierungsrichtlinie basierend auf Amazon SQS	461
Eine Skalierung überprüfen	469
Eine Skalierungsrichtlinie deaktivieren	472
Löschen Sie eine Skalierungsrichtlinie für eine Auto Scaling Scaling-Gruppe	474
AWS CLI Beispiele für Skalierungsrichtlinien	477
Prädiktive Skalierung	480
So funktioniert Auto Scaling	481

Eine Richtlinie für die prädiktive Skalierung erstellen	486
Auswertung Ihrer Richtlinien für prädiktive Skalierung	494
Prognose überschreiben	504
Verwenden benutzerdefinierter Metriken	510
Instance-Beendigung steuern	522
Szenarien für Beendigungsrichtlinien	523
Kündigungsrichtlinien konfigurieren	527
Eine benutzerdefinierte Beendigungsrichtlinie mit Lambda erstellen	533
Instance-Abskalierungsschutz verwenden	541
Sorgen Sie für eine ordnungsgemäße Instance-Beendigung	545
Aussetzen und Fortsetzen von Prozessen	554
Arten von Prozessen	555
Überlegungen	556
Prozess anhalten	557
Prozesse fortsetzen	558
Wie sich unterbrochene Prozesse auf andere Prozesse auswirken	559
Überwachen	563
Health checks (Zustandsprüfungen)	565
Über Zustandsprüfungen	567
Legen Sie Nachfrist für Zustandsprüfungen fest	574
Überwachen Sie, ob Amazon EBS-Volumen beeinträchtigt sind	577
Richten Sie einen benutzerdefinierten Integritätscheck ein	580
Anzeigen des Grundes für Fehler bei Zustandsprüfung	582
Problembhebung bei instabilen Instances	584
Überwachen Sie mit Health Dashboard	587
Überwachen Sie CloudWatch Messwerte	589
Überwachungsgrafiken in der Amazon EC2 Auto Scaling-Konsole anzeigen	589
CloudWatch Metriken für Amazon EC2 Auto Scaling	594
Überwachung für Auto-Scaling-Instances konfigurieren	603
API-Aufrufe protokollieren mit CloudTrail	606
Auto Scaling Scaling-Verwaltungsereignisse in CloudTrail	607
Beispiele Auto Scaling Scaling-Ereignisse	607
Auto Scaling RemoveAction ruft auf CloudWatch	609
Amazon SNS SNS-Benachrichtigungsoptionen	609
Amazon SNS und Amazon EC2 Auto Scaling	610
Arbeiten mit anderen Services	617

Kapazitätsausgleich	617
-Übersicht	618
Verhalten bei Kapazitätswiederherstellungen	619
Überlegungen	620
Aktivieren Sie den Kapazitätsneuausgleich.	622
Kapazitätsreservierungen	629
Einstellung „Kapazitätsreservierung“	629
Balance zwischen Verfügbarkeitszonen und Kapazitätsreservierungen	631
Verwenden Sie die Präferenz Kapazitätsreservierung	631
Reservierungen unterbrechbarer Kapazitäten	633
AWS CloudShell	638
CloudFormation	639
Amazon EC2 Auto Scaling und Vorlagen CloudFormation	639
Erfahren Sie mehr über CloudFormation	640
Compute Optimizer	640
Einschränkungen	641
Ergebnisse	641
Anzeigen von Empfehlungen	642
Überlegungen zur Bewertung der Empfehlungen	643
Elastic Load Balancing	644
Arten von Elastic Load Balancing	645
Bereiten Sie sich darauf vor, einen Load Balancer anzuschließen	647
Hinzufügen eines Load Balancers	650
Konfigurieren Sie einen Load Balancer	654
Überprüfen des Anhangsstatus	655
Fügen Sie eine Availability Zone hinzu	656
Entfernen einer Availability Zone	659
Entfernen eines Load Balancers	653
AWS CLI Beispiele für die Arbeit mit Elastic Load Balancing	661
VPC-Gitter	669
Vorbereitung auf das Hinzufügen einer Zielgruppe	671
Fügen Sie eine VPC-Lattice-Zielgruppe hinzu	674
Überprüfen des Anhangsstatus	679
EventBridge	681
Ereignis-Referenz für Amazon EC2 Auto Scaling	681
Beispiele für Ereignisse und Muster für die Aktualisierung von Instanzen	688

Beispielereignisse und -muster in einem warmen Pool	696
EventBridge Regeln	701
Amazon VPC	707
Standard-VPC	708
Nicht standardmäßige VPC	708
Überlegungen bei der Auswahl von VPC-Subnetzen	708
IP-Adressierung in einer VPC	709
Netzwerkschnittstellen in einer VPC	710
Tenancy zur Instance-Platzierung	710
AWS Outposts	710
Weitere Ressourcen, um mehr zu erfahren über VPCs	711
Sicherheit	712
Sicherheit der Infrastruktur	713
Zugehörige Ressourcen	713
Ausfallsicherheit	713
Zugehörige Ressourcen	715
Datenschutz	715
Wird AWS KMS keys zum Verschlüsseln von Amazon EBS-Volumes verwendet	716
Zugehörige Ressourcen	717
AWS KMS wichtige Richtlinie für die Verwendung mit verschlüsselten Volumes	717
Identitäts- und Zugriffsverwaltung	724
Zugriffskontrolle	725
Funktionsweise von Amazon EC2 Auto Scaling mit IAM	725
API-Berechtigungen	734
Verwaltete Richtlinien	736
Service-verknüpfte Rollen	742
Beispiele für identitätsbasierte Richtlinien	747
Serviceübergreifende Confused-Deputy-Prävention	758
Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling- Gruppen	760
IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden	769
Compliance-Validierung	773
Compliance mit PCI DSS	773
Verwenden Sie VPC-Endpunkte für private Konnektivität	774
Erstellen eines Schnittstellen-VPC-Endpunkts	774
Erstellen einer VPC-Endpunktrichtlinie	774

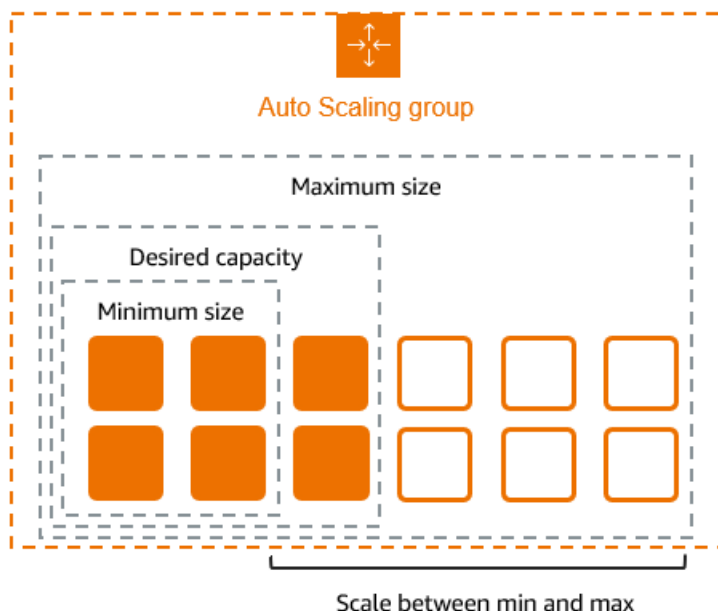
Arbeitet mit AWS SDKs	776
Codebeispiele	778
Grundlagen	780
Hello Auto Scaling	782
Kennenlernen der Grundlagen	792
Aktionen	894
Szenarien	1128
Erstellen und Verwalten eines ausfallsicheren Services	1128
Fehlerbehebung	1298
Abrufen einer Fehlermeldung	1298
Schalten Sie Skalierungsaktivitäten aus	1300
Weitere Ressourcen zur Fehlerbehebung	1301
Fehler beim Starten von Instances	1302
Die angefragte Konfiguration wird derzeit nicht unterstützt.	1303
Die Sicherheitsgruppe <Name der Sicherheitsgruppe> ist nicht vorhanden. Die EC2- Instance konnte nicht gestartet werden.	1304
Das Schlüsselpaar <mit Ihrer EC2-Instance verbundenes Schlüsselpaar> ist nicht vorhanden. Die EC2-Instance konnte nicht gestartet werden.	1304
Der von Ihnen angeforderte Instance-Typ (<Instance type>) wird in der von Ihnen angeforderten Availability Zone (<instance Availability Zone>) nicht unterstützt... ..	1305
Ihr Spot-Anfragepreis von 0,015 ist niedriger als der erforderliche Mindestpreis für Spot- Anfragen von 0,0735... ..	1305
Ungültiger Gerätenamenname <Gerätenamenname> / Ungültiger Gerätenamenname beim Hochladen. Die EC2- Instance konnte nicht gestartet werden.	1306
Der Wert (<Name des verbundenen Instance-Speichergeräts>) für den Parameter virtualName ist ungültig... Die EC2-Instance konnte nicht gestartet werden.	1306
EBS-Blockgerätezuordnungen werden für den Instance-Store nicht unterstützt. AMIs	1307
Platzierungsgruppen dürfen nicht mit Instanzen des Typs '<Instance type>' verwendet werden. Die EC2-Instance konnte nicht gestartet werden.	1307
Kunde. InternalError: Client-Fehler beim Start.	1307
Wir haben derzeit nicht genügend <instance type>-Kapazität in der Availability Zone, die Sie angefragt haben. Die EC2-Instance konnte nicht gestartet werden.	1309
Die angefragte Reservierung ist nicht ausreichend kompatibel und hat nicht genügend freie Kapazität für diese Anfrage. Die EC2-Instance konnte nicht gestartet werden.	1310
Ihre Kapazitätsblock-Reservierung <reservation id> ist noch nicht aktiv. Die EC2-Instance konnte nicht gestartet werden.	1310

Es ist keine Spot-Kapazität verfügbar, die Ihrer Anforderung entspricht. Die EC2-Instance konnte nicht gestartet werden.	1311
<number of instances> Instance wird/Instances werden bereits ausgeführt. Die EC2-Instance konnte nicht gestartet werden.	1311
AMI-Probleme	1312
Die AMI-ID <ID of your AMI> existiert nicht. Die EC2-Instance konnte nicht gestartet werden.	1312
Das AMI <AMI-ID> hat den Status "Schwebend" und kann nicht ausgeführt werden. Die EC2-Instance konnte nicht gestartet werden.	1313
Ungültiger Gerätenamenname <device name>. Die EC2-Instance konnte nicht gestartet werden.	1313
Die Architektur 'arm64' des angegebenen Instance-Typs entspricht nicht der Architektur 'x86_64' des angegebenen AMI... Das Starten der EC2-Instance ist fehlgeschlagen.	1313
AMI '<AMI ID>' ist deaktiviert und kann nicht ausgeführt werden. Die EC2-Instance konnte nicht gestartet werden.	1315
Load Balancer-Probleme	1315
Eine oder mehrere Zielgruppen. Das Validieren der Load Balancer-Konfiguration ist fehlgeschlagen.	1316
Load Balancer kann nicht gefunden <your load balancer>werden. Das Validieren der Load Balancer-Konfiguration ist fehlgeschlagen.	1317
Es ist kein AKTIVER Load Balancer namens <Load Balancer-Name> vorhanden. Das Aktualisieren der Load Balancer-Konfiguration ist fehlgeschlagen.	1317
Die EC2-Instance <instance ID> ist nicht in der VPC. Das Aktualisieren der Load Balancer-Konfiguration ist fehlgeschlagen.	1318
Startvorlagenprobleme	1318
Sie müssen eine gültige, vollständig formatierte Startvorlage verwenden (ungültiger Wert)	1318
Sie sind nicht berechtigt, die Startvorlage zu verwenden (unzureichende Berechtigungen)	1319
Ähnliche Informationen	1321
Dokumentverlauf	1323
.....	mccclxx

Was ist Amazon EC2 Auto Scaling?

Amazon EC2 Auto Scaling hilft Ihnen sicherzustellen, dass Sie die richtige Anzahl von Amazon EC2-Instances zur Verfügung haben, um die Auslastung Ihrer Anwendung zu bewältigen. Sie erstellen Sammlungen von EC2 Instances, die als Auto Scaling-Gruppen bezeichnet werden. Sie können die minimale Anzahl von Instances in jeder Auto Scaling-Gruppe angeben und Amazon EC2 Auto Scaling stellt sicher, dass Ihre Gruppe nie kleiner als diese Grösse wird. Sie können die maximale Anzahl von Instances in jeder Auto-Scaling-Gruppe angeben und Amazon EC2 Auto Scaling stellt sicher, dass Ihre Gruppe nie grösser als diese Grösse wird. Wenn Sie die gewünschte Kapazität angeben – entweder beim Erstellen der Gruppe oder später – stellt Amazon EC2 Auto Scaling sicher, dass Ihrer Gruppe die entsprechende Anzahl an Instances zugewiesen wird. Wenn Sie Skalierungsrichtlinien angeben, kann Amazon EC2 Auto Scaling je nach Bedarf Ihrer Anwendung Instances starten oder beenden.

Die folgende Auto Scaling Scaling-Gruppe hat beispielsweise eine Mindestgröße von vier Instances, eine gewünschte Kapazität von sechs Instances und eine Maximalgröße von zwölf Instances. Die Anzahl an Instances wird gemäß den von Ihnen festgelegten Skalierungsrichtlinien angepasst. Sie liegt immer zwischen Ihrer Mindest- und Höchstanzahl an Instances und richtet sich nach den von Ihnen angegebenen Kriterien.



Funktionen von Amazon EC2 Auto Scaling

Mit Amazon EC2 Auto Scaling sind Ihre EC2-Instances in Auto Scaling-Gruppen organisiert, sodass sie für Skalierungs- und Verwaltungszwecke als logische Einheit behandelt werden können. Auto Scaling Scaling-Gruppen verwenden Startvorlagen (oder Startkonfigurationen) als Konfigurationsvorlagen für ihre EC2-Instances.

Im Folgenden sind die wichtigsten Funktionen von Amazon EC2 Auto Scaling aufgeführt:

Überwachung des Zustands laufender Instances

Amazon EC2 Auto Scaling überwacht automatisch den Zustand und die Verfügbarkeit Ihrer Instances mithilfe von EC2-Zustandsprüfungen und ersetzt beendete oder beeinträchtigte Instances, um die gewünschte Kapazität aufrechtzuerhalten.

Benutzerdefinierte Zustandsprüfungen

Zusätzlich zu den integrierten Integritätsprüfungen können Sie benutzerdefinierte Zustandsprüfungen definieren, die speziell auf Ihre Anwendung zugeschnitten sind, um sicherzustellen, dass sie erwartungsgemäß reagiert. Wenn eine Instance Ihre benutzerdefinierte Zustandsprüfung nicht besteht, wird sie automatisch ersetzt, um die gewünschte Kapazität aufrechtzuerhalten.

Kapazitätsausgleich zwischen Availability Zones

Sie können mehrere Availability Zones für Ihre Auto Scaling-Gruppe angeben, und Amazon EC2 Auto Scaling verteilt Ihre Instances gleichmäßig auf die Availability Zones, wenn die Gruppe skaliert. Dies sorgt für hohe Verfügbarkeit und Resilienz, indem Ihre Anwendungen an einem einzigen Standort vor Ausfällen geschützt werden.

Mehrere Instance-Typen und Kaufoptionen

Innerhalb einer einzigen Auto Scaling Scaling-Gruppe können Sie mehrere Instance-Typen und Kaufoptionen (Spot- und On-Demand-Instances) starten, sodass Sie die Kosten durch die Nutzung von Spot-Instances optimieren können. Sie können auch Rabatte für Reserved Instances und Savings Plans nutzen, indem Sie sie in Verbindung mit On-Demand-Instances in der Gruppe verwenden.

Automatischer Ersatz von Spot Instances

Wenn Ihre Gruppe Spot-Instances umfasst, kann Amazon EC2 Auto Scaling automatisch Ersatz-Spot-Kapazität anfordern, falls Ihre Spot-Instances unterbrochen werden. Durch Capacity

Rebalancing kann Amazon EC2 Auto Scaling auch Ihre Spot-Instances, bei denen ein erhöhtes Ausfallrisiko besteht, überwachen und proaktiv ersetzen.

Lastausgleich

Mit Elastic Load Balancing Load Balancing und Health Checks können Sie sicherstellen, dass der Anwendungsdatenverkehr gleichmäßig auf Ihre intakten Instances verteilt wird. Immer wenn Instances gestartet oder beendet werden, registriert Amazon EC2 Auto Scaling die Instances automatisch und meldet sie vom Load Balancer ab.

Skalierbarkeit

Amazon EC2 Auto Scaling bietet Ihnen auch mehrere Möglichkeiten, Ihre Auto Scaling-Gruppen zu skalieren. Mithilfe von Auto Scaling können Sie die Anwendungsverfügbarkeit aufrechterhalten und die Kosten senken, indem Sie Kapazität hinzufügen, um Lastspitzen zu bewältigen, und Kapazität entfernen, wenn der Bedarf geringer ist. Sie können die Größe Ihrer Auto Scaling Scaling-Gruppe nach Bedarf auch manuell anpassen.

Instance-Aktualisierung

Die Instance-Aktualisierungsfunktion bietet einen Mechanismus, um Instances fortlaufend zu aktualisieren, wenn Sie Ihr AMI oder Ihre Startvorlage aktualisieren. Sie können auch einen schrittweisen Ansatz verwenden, der als Canary-Deployment bezeichnet wird, um ein neues AMI oder eine neue Startvorlage auf einer kleinen Anzahl von Instances zu testen, bevor Sie es für die gesamte Gruppe bereitstellen.

Lebenszyklus-Hooks

Lifecycle-Hooks sind nützlich, um benutzerdefinierte Aktionen zu definieren, die beim Start neuer Instances oder vor dem Beenden von Instances aufgerufen werden. Diese Funktion ist besonders nützlich für den Aufbau ereignisgesteuerter Architekturen, hilft Ihnen aber auch dabei, Instanzen während ihres gesamten Lebenszyklus zu verwalten.

Support für statusbehaftete Workloads

Lifecycle-Hooks bieten auch einen Mechanismus, um den Status beim Herunterfahren beizubehalten. Um die Kontinuität von statusbehafteten Anwendungen zu gewährleisten, können Sie auch Scale-in-Schutz oder benutzerdefinierte Kündigungsrichtlinien verwenden, um zu verhindern, dass Instances mit lang andauernden Prozessen vorzeitig beendet werden.

Weitere Informationen über die Vorteile von Amazon EC2 Auto Scaling finden Sie unter [Vorteile von Auto Scaling für die Anwendungsarchitektur](#).

Preise für Amazon EC2 Auto Scaling

Bei Amazon EC2 Auto Scaling fallen keine zusätzlichen Gebühren an. Sie können es also ganz einfach ausprobieren und herausfinden, wie es Ihrer AWS Architektur zugute kommen kann. Sie zahlen nur für die AWS Ressourcen (z. B. EC2-Instances, EBS-Volumes und CloudWatch Alarme), die Sie tatsächlich nutzen.

Erste Schritte

Schließen Sie zunächst das Tutorial [Erstellen Sie Ihre erste Auto Scaling Scaling-Gruppe](#) ab, um eine Auto Scaling Scaling-Gruppe zu erstellen und zu sehen, wie sie reagiert, wenn eine Instance in dieser Gruppe beendet wird.

Arbeiten mit Auto-Scaling-Gruppen

Sie können die folgenden Schnittstellen verwenden, um Ihre Auto-Scaling-Gruppen zu erstellen, auf sie zuzugreifen und sie zu verwalten:

- **AWS-Managementkonsole** – Bietet eine Webschnittstelle für den Zugriff auf Ihre Auto-Scaling-Gruppen. Wenn Sie sich für eine angemeldet haben AWS-Konto, können Sie auf Ihre Auto Scaling Scaling-Gruppen zugreifen, indem Sie sich bei der anmelden AWS-Managementkonsole, das Suchfeld in der Navigationsleiste verwenden, um nach Auto Scaling Scaling-Gruppen zu suchen, und dann Auto Scaling Scaling-Gruppen auswählen.
- **AWS Command Line Interface (AWS CLI)** — Stellt Befehle für eine Vielzahl von AWS-Services Befehlen bereit und wird unter Windows, MacOS und Linux unterstützt. Um zu beginnen, sehen Sie sich [Bereiten Sie sich auf die Nutzung des vor AWS CLI](#) an. Weitere Informationen finden Sie unter [autoscaling](#) in der AWS CLI -Befehlsreferenz.
- **AWS Tools for Windows PowerShell**— Stellt Befehle für eine breite Palette von AWS Produkten für Benutzer bereit, die in der PowerShell Umgebung Skripts erstellen. Informationen zu den ersten Schritten finden Sie im [AWS -Tools für PowerShell -Benutzerhandbuch](#). Weitere Informationen finden Sie in der [AWS -Tools für PowerShell Cmdlet-Referenz](#).
- **AWS SDKs**— Stellt sprachspezifische API-Operationen bereit und kümmert sich um viele Verbindungsdetails, wie z. B. die Berechnung von Signaturen, die Behandlung von Wiederholungsversuchen von Anfragen und die Behandlung von Fehlern. Weitere Informationen finden Sie unter [AWS SDKs](#).

- Abfrage-API – Bietet API-Aktionen auf niedriger Ebene, die Sie mithilfe von HTTPS-Anforderungen aufrufen. Die Verwendung der Abfrage-API ist die direkteste Möglichkeit für den Zugriff auf AWS-Services. Allerdings müssen dann viele technische Abläufe, wie beispielsweise das Erzeugen des Hashwerts zum Signieren der Anforderung und zur Fehlerbehandlung, in der Anwendung durchgeführt werden. Weitere Informationen finden Sie in der [Amazon EC2 Auto Scaling-API-Referenz](#).
- CloudFormation— Unterstützt das Erstellen von Auto Scaling Scaling-Gruppen mithilfe von CloudFormation Vorlagen. Weitere Informationen finden Sie unter [Erstellen Sie Auto Scaling Scaling-Gruppen mit AWS CloudFormation](#).

Um programmgesteuert eine Verbindung zu einem herzustellen AWS-Service, verwenden Sie einen Endpunkt.

Vorteile von Auto Scaling für die Anwendungsarchitektur

Das Hinzufügen von Amazon EC2 Auto Scaling zu Ihrer Anwendungsarchitektur ist eine Möglichkeit, die Vorteile der AWS Cloud zu maximieren. Wenn Sie Amazon EC2 Auto Scaling verwenden, profitieren Sie bei Ihren Anwendungen von folgenden Vorteilen:

- Bessere Fehlertoleranz. Amazon EC2 Auto Scaling erkennt, wenn eine Instance fehlerhaft ist, beendet sie und startet eine andere Instance, um die fehlerhafte zu ersetzen. Sie können Amazon EC2 Auto Scaling auch so konfigurieren, dass mehrere Availability Zones genutzt werden. Wenn eine Availability Zone ausfällt, startet Amazon EC2 Auto Scaling die Instances in einer anderen, um den Ausfall zu kompensieren.
- Bessere Verfügbarkeit. Mit Amazon EC2 Auto Scaling können Sie sicherstellen, dass für Ihre Anwendung immer genügend Kapazität zur Verarbeitung des aktuellen Datenverkehrs verfügbar ist.
- Bessere Kostenkontrolle. Amazon EC2 Auto Scaling passt die Kapazität je nach Bedarf dynamisch an. Da Sie für die Nutzung von EC2-Instances zahlen, sparen Sie Kosten, wenn Sie sie nur bei Bedarf starten und wieder beenden, sobald Sie sie nicht mehr brauchen.

Inhalt

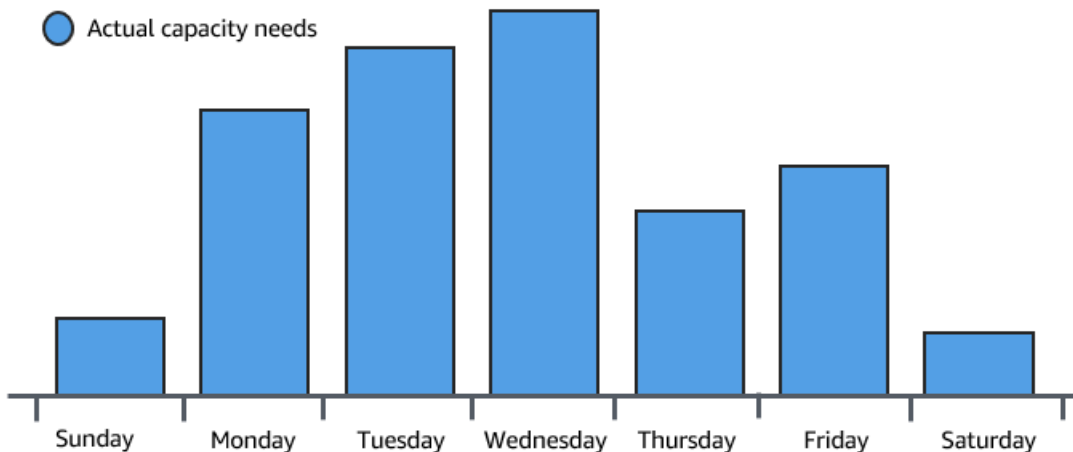
- [Beispiel: Abdecken des Variablenbedarfs](#)
- [Beispiel: Architektur für eine Web-App](#)
- [Beispiel: Aufteilen von Instances in mehrere Availability Zones](#)

- [Instance-Distribution](#)
- [Wiederherstellen des Gleichgewichts von Aktivitäten](#)

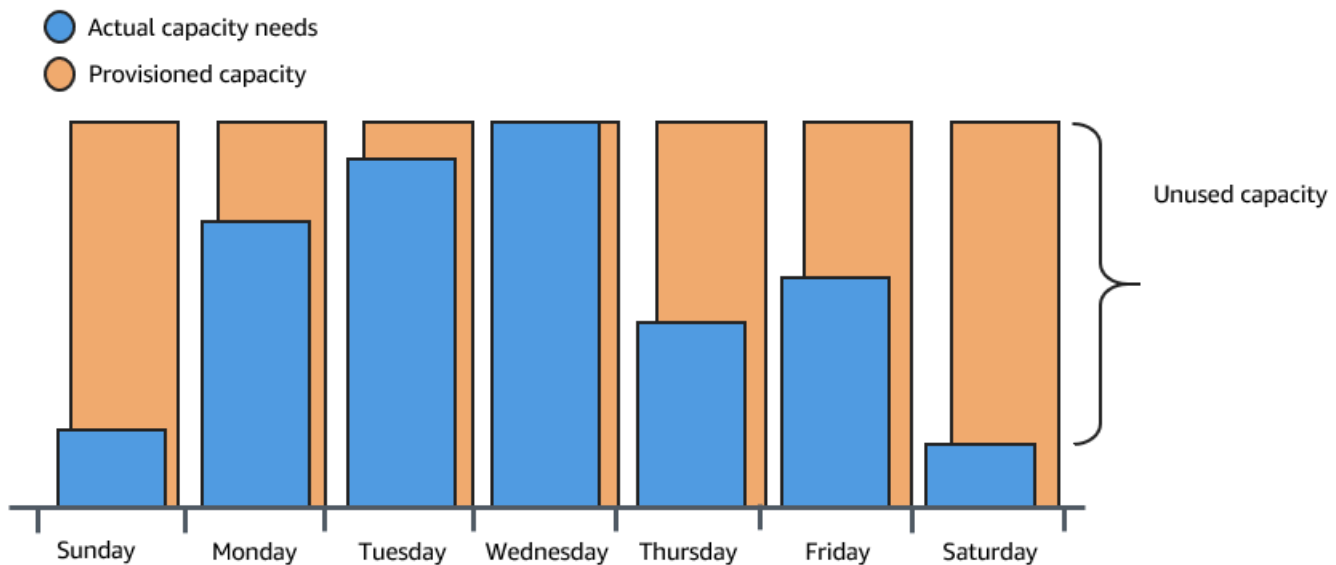
Beispiel: Abdecken des Variablenbedarfs

Wir möchten Ihnen einige der Vorteile von Amazon EC2 Auto Scaling anhand einer einfachen Webanwendung zeigen, die in AWS ausgeführt wird. Mit dieser Anwendung können Mitarbeiter Konferenzräume für Meetings suchen. Am Anfang und am Ende der Woche wird diese Anwendung nur wenig genutzt. Mitte der Woche planen mehr Mitarbeiter ihre Meetings, die Anforderungen an die Anwendung erhöhen sich also deutlich.

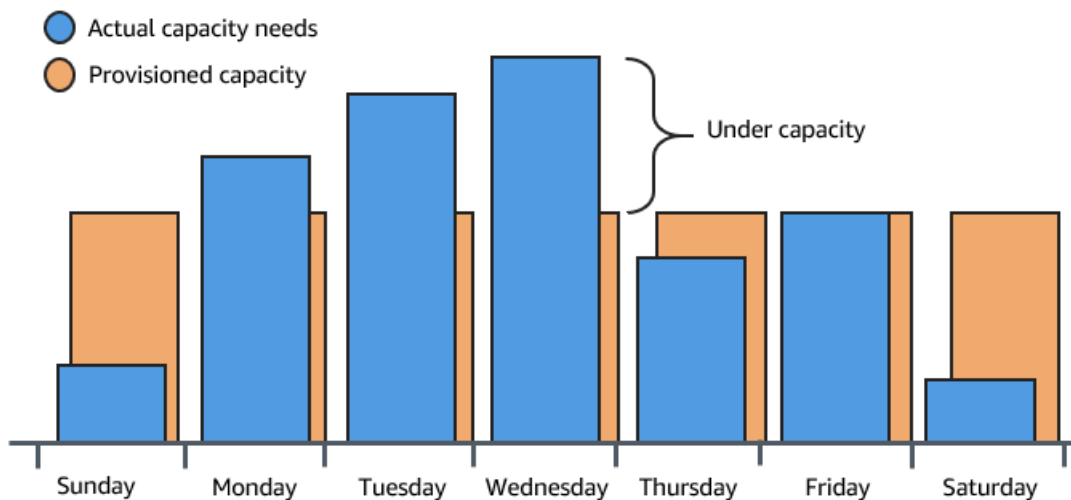
Das folgende Diagramm zeigt, wie viel Kapazität der Anwendung im Lauf der Woche verwendet wird.



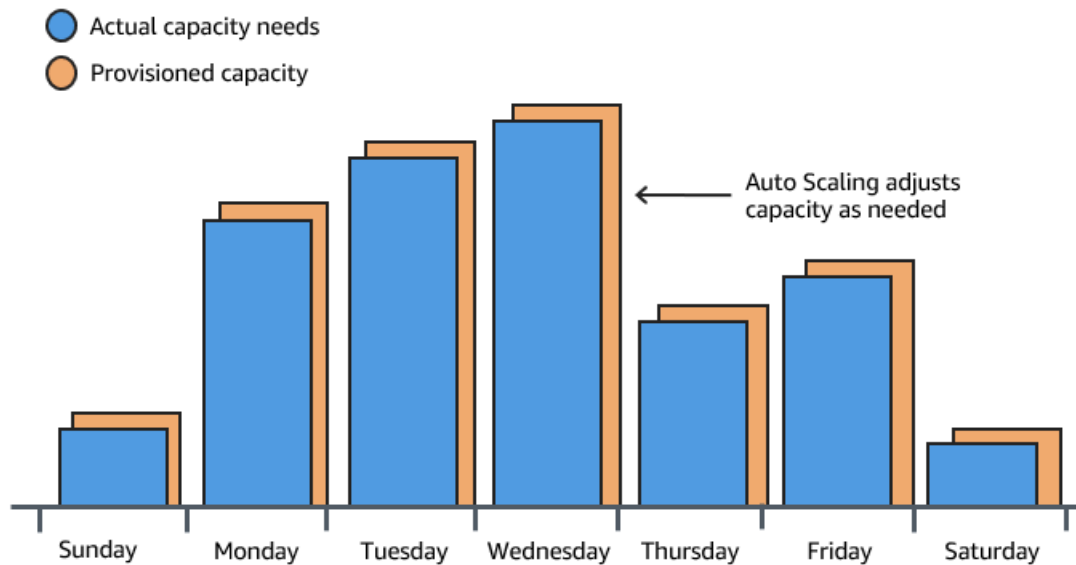
Bisher gab es zwei nur Möglichkeiten, diese Änderungen in der Kapazitätsplanung zu berücksichtigen. Bei der ersten Möglichkeit werden so viele Server hinzugefügt, dass der Kapazitätsbedarf der Anwendung immer gedeckt ist. Diese Möglichkeit hat jedoch einen Nachteil. An machen Tagen benötigt die Anwendung nicht so viel Kapazität. Die zusätzliche Kapazität bleibt ungenutzt und erhöht im Grunde genommen die Kosten für das Ausführen der Anwendung.



Bei der zweiten Möglichkeit wird darauf geachtet, dass die Kapazität für den durchschnittlichen Bedarf der Anwendung ausreicht. Diese Möglichkeit ist kostengünstiger, da Sie keine Geräte kaufen, die Sie nur gelegentlich verwenden. Allerdings riskieren Sie eine schlechte Kundenerfahrung, wenn der Bedarf der Anwendung die Kapazität übersteigt.



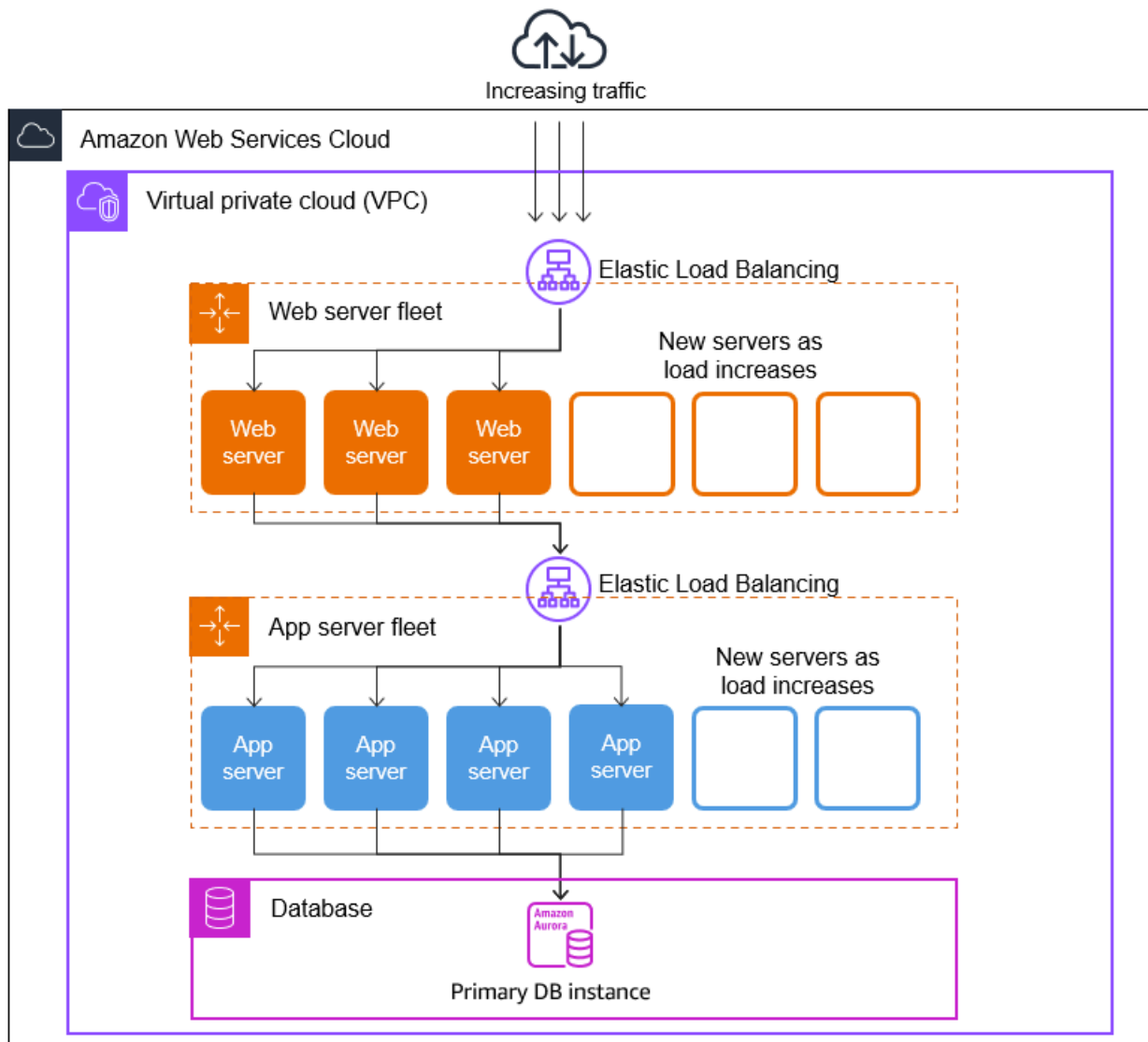
Wenn Sie dieser Anwendung Amazon EC2 Auto Scaling hinzufügen, bietet sich eine dritte Möglichkeit. Sie können der Anwendung bei Bedarf neue Instances hinzufügen und diese wieder beenden, wenn Sie sie nicht mehr benötigen. Da Amazon EC2 Auto Scaling EC2-Instances verwendet, bezahlen Sie nur für die Instances, die Sie tatsächlich nutzen. Sie verfügen jetzt über eine kosteneffektive Architektur, die eine optimale Kundenerfahrung erzielt und gleichzeitig die Kosten gering hält.



Beispiel: Architektur für eine Web-App

Bei den meisten Web-Apps werden mehrere Kopien der App gleichzeitig ausgeführt, um dem Kunden-Datenverkehr gerecht zu werden. Die einzelnen Kopien Ihrer Anwendung werden auf identischen EC2-Instances (Cloud-Servern) gehostet, die die Kundenanfragen jeweils bearbeiten.

Amazon EC2 Auto Scaling verwaltet das Starten und Beenden dieser EC2-Instances für Sie. Sie definieren eine Reihe von Kriterien (z. B. einen CloudWatch Amazon-Alarm), die bestimmen, wann die Auto Scaling Scaling-Gruppe EC2-Instances startet oder beendet. Das Hinzufügen von Auto-Scaling-Gruppen zur Ihrer Netzwerkarchitektur trägt dazu bei, dass sich die Verfügbarkeit und Fehlertoleranz Ihrer Anwendung verbessern.



Sie können so viele Auto-Scaling-Gruppen erstellen, wie Sie benötigen. Sie können beispielsweise für jede Ebene eine Auto-Scaling-Gruppe erstellen.

Damit der Datenverkehr zwischen den Instances in Ihren Auto-Scaling-Gruppen aufgeteilt wird, können Sie einen Load Balancer in Ihre Architektur aufnehmen. Weitere Informationen finden Sie unter [Elastic Load Balancing](#).

Beispiel: Aufteilen von Instances in mehrere Availability Zones

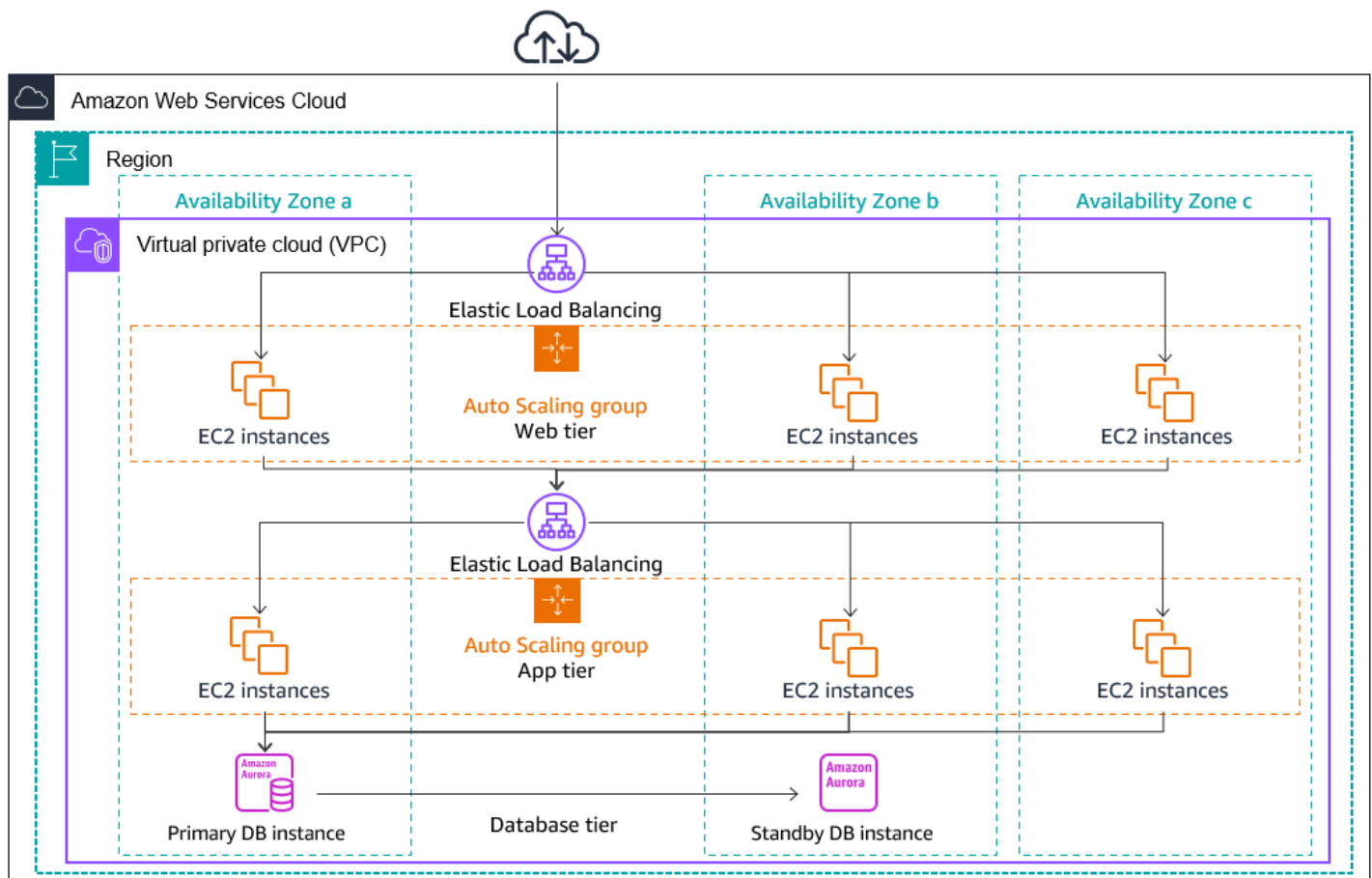
Availability Zones sind isolierte Standorte innerhalb in einer gegebenen AWS-Region-Region. Jede Region verfügt über mehrere Availability Zones, die eine hohe Verfügbarkeit für die Region bieten. Availability Zones sind unabhängig, und daher erhöhen Sie die Anwendungsverfügbarkeit, wenn Sie

Ihre Anwendung so entwerfen, dass sie mehrere Zonen verwendet. Weitere Informationen finden Sie unter [Ausfallsicherheit in Amazon EC2 Auto Scaling](#).

Eine Availability Zone wird durch den AWS-Region Code gefolgt von einer Buchstabenkennung identifiziert (z. B. us-east-1a). Wenn Sie die VPC und Subnetze erstellen, anstatt die Standard-VPC zu verwenden, können Sie eines oder mehrere Subnetze in jeder Availability Zone definieren. Jedes Subnetz muss sich vollständig innerhalb einer Availability Zone befinden und darf nicht mehrere Zonen umfassen. Weitere Informationen finden Sie unter [Wie funktioniert Amazon VPC](#) im Benutzerhandbuch für Amazon VPC.

Wenn Sie eine Auto-Scaling-Gruppe erstellen, müssen Sie die VPC und die Subnetze auswählen, in denen Sie die Auto-Scaling-Gruppe bereitstellen möchten. Amazon EC2 Auto Scaling erstellt Ihre Instances in den von Ihnen ausgewählten Subnetzen. Jede Instance ist somit einer bestimmten Availability Zone zugeordnet, die von Amazon EC2 Auto Scaling ausgewählt wurde. Wenn Instances gestartet werden, versucht Amazon EC2 Auto Scaling, sie gleichmäßig auf die Zonen zu verteilen, um eine hohe Verfügbarkeit und Zuverlässigkeit zu gewährleisten.

Sie sehen hier einen Überblick über die mehrstufige Architektur, die in drei Availability Zones eingesetzt wird.



Instance-Distribution

Amazon EC2 Auto Scaling versucht automatisch, die gleiche Anzahl von Instances in jeder aktivierten Availability Zone aufrechtzuerhalten. Amazon EC2 Auto Scaling versucht dazu, in der Availability Zone mit den wenigsten Instances neue Instances zu starten. Wenn mehrere Subnetze für die Availability Zone ausgewählt wurden, versucht Amazon EC2 Auto Scaling, Instances in dem Subnetz zu starten, das die höchste Anzahl an verfügbaren IP-Adressen in der Availability Zone hat. Falls dies fehlschlägt, versucht Amazon EC2 Auto Scaling so lange, die Instances in einer anderen Availability Zone zu starten, bis dies gelingt.

Unter Umständen, in denen eine Availability Zone nicht mehr funktioniert oder nicht mehr verfügbar ist, kann die Verteilung der Instances auf die Availability Zones ungleichmäßig verteilt werden. Wenn die Availability Zone wiederhergestellt ist, gleicht Amazon EC2 Auto Scaling die Auto-Scaling-Gruppe automatisch neu aus. Dies geschieht, indem Instances in den aktivierten Availability Zones mit den wenigsten Instances gestartet und Instances an anderer Stelle beendet werden.

Wiederherstellen des Gleichgewichts von Aktivitäten

Neuausgleichsaktivitäten fallen in zwei Kategorien: Neuausgleich der Availability Zone und Neuausgleich der Kapazität.

Neuausgleich der Availability Zone

Nach bestimmten Aktionen kann das Verhältnis der Availability Zones Ihrer Auto-Scaling-Gruppe aus dem Gleichgewicht geraten. Amazon EC2 Auto Scaling kann dies kompensieren und das Gleichgewicht der Availability Zones wiederherstellen. Folgende Aktionen können ein Wiederherstellen des Gleichgewichts erforderlich machen:

- Sie ändern die Availability Zones, die Ihrer Auto-Scaling-Gruppe zugeordnet sind.
- Sie beenden oder trennen Instances explizit, oder versetzen Instances in den Standby-Modus, wodurch die Gruppe aus dem Gleichgewicht gerät.
- Eine Availability Zone, die bisher zu wenig Kapazität hatte, wurde wiederhergestellt, wodurch jetzt zusätzliche Kapazität zur Verfügung steht.
- Eine Availability Zone mit einem Spot-Preis, der bisher über Ihrem Höchstpreis lag, liegt jetzt darunter.

Beim Wiederherstellen des Gleichgewichts von Instances startet Amazon EC2 Auto Scaling neue Instances, bevor die alten beendet werden. Auf diese Weise beeinträchtigt ein Wiederherstellen des Gleichgewichts die Leistung und Verfügbarkeit Ihrer Anwendung nicht.

Da Amazon EC2 Auto Scaling vor dem Beenden der vorherigen Instances versucht, neue zu starten, kann das Wiederherstellen des Gleichgewichts beeinträchtigt und sogar gänzlich unterbrochen werden, falls die angegebene maximale Kapazität nahezu oder gänzlich erreicht ist.

Um dieses Problem zu vermeiden, kann das System beim Wiederherstellen des Gleichgewichts die angegebene maximale Kapazität einer Gruppe vorübergehend überschreiten. Standardmäßig kann dies mit einer Marge von 10 Prozent oder einer Instance geschehen, je nachdem, welcher Wert größer ist. Die Marge wird nur verlängert, wenn die Gruppe die maximale Kapazität erreicht oder fast erreicht und eine Neugewichtung erforderlich ist. Die Kapazität wird nur für die Dauer der Wiederherstellung des Gleichgewichts in der Gruppe erhöht, in der Regel sind dies einige Minuten.

Alternativ können Sie mithilfe einer Wartungsrichtlinie für Instances Schwellenwerte für eine Auto-Scaling-Gruppe festlegen, und die Gruppe kann die Kapazität nur innerhalb dieses Schwellenwertbereichs erhöhen oder verringern. Auf diese Weise können Sie kontrollieren, wie

schnell Ihre Gruppe eine erneute Verteilung durchführt. Weitere Informationen finden Sie unter [Wartungsrichtlinien für Instances](#).

Kapazitätsausgleich

Wenn Sie Spot-Instances verwenden, können Sie den Kapazitätsausgleich für Ihre Auto-Scaling-Gruppen aktivieren. Dadurch kann Amazon EC2 Auto Scaling versuchen, eine Spot-Instance zu starten, wenn Amazon EC2 benachrichtigt, dass eine Spot-Instance einem erhöhten Unterbrechungsrisiko ausgesetzt ist. Nach dem Start einer neuen Instance wird dann eine frühere Instance beendet. Weitere Informationen finden Sie unter [Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances](#).

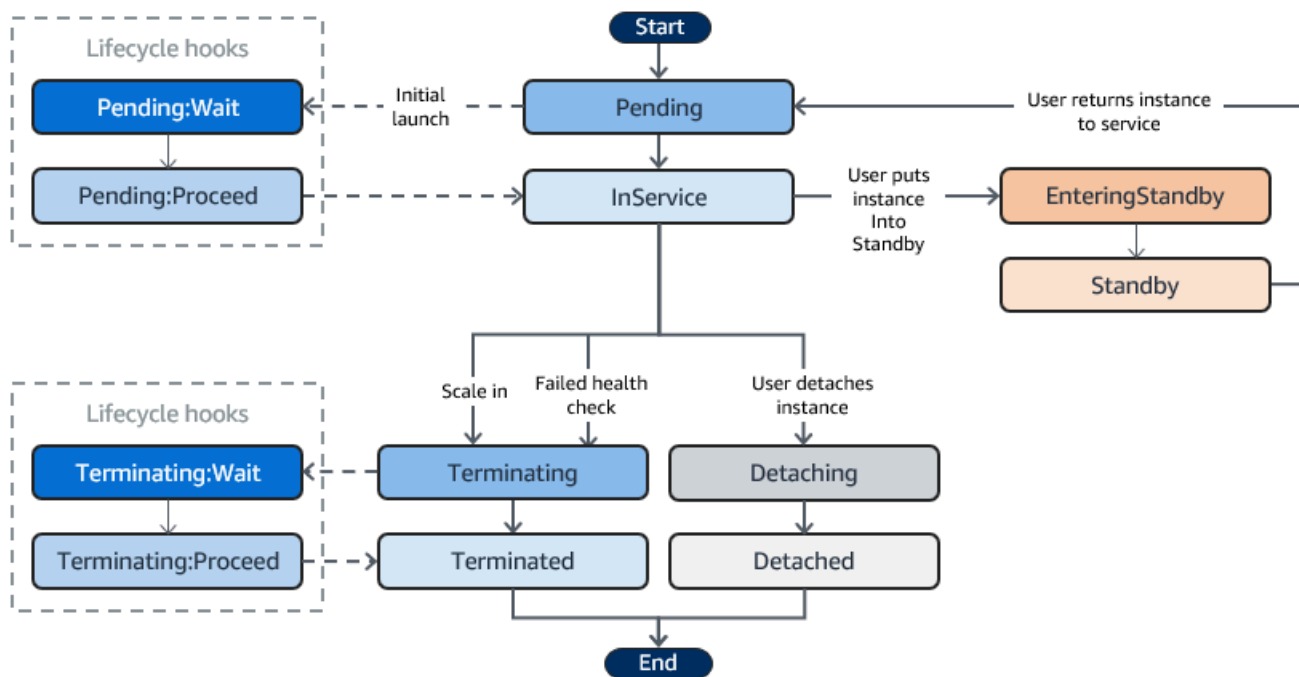
Instance-Lebenszyklus bei Amazon EC2 Auto Scaling

Der Lebenszyklus der EC2-Instances in einer Auto-Scaling-Gruppe unterscheidet sich von dem anderer EC2-Instances. Der Lebenszyklus beginnt, wenn die Auto-Scaling-Gruppe eine Instance startet und sie in Betrieb nimmt. Der Lebenszyklus endet, wenn Sie die Instance beenden oder die Auto-Scaling-Gruppe die Instance ausmustert und beendet.

Note

Instances werden Ihnen in Rechnung gestellt, sobald sie gestartet werden, einschließlich der Zeit, die sie noch nicht in Betrieb sind.

Die folgende Abbildung zeigt die Übergänge zwischen den Instance-Status im Amazon EC2 Auto Scaling-Lebenszyklus.



Horizontale Skalierung

Die folgenden horizontalen Skalierungsereignisse weisen die Auto-Scaling-Gruppe an, EC2-Instances zu starten und sie der Gruppe anzufügen:

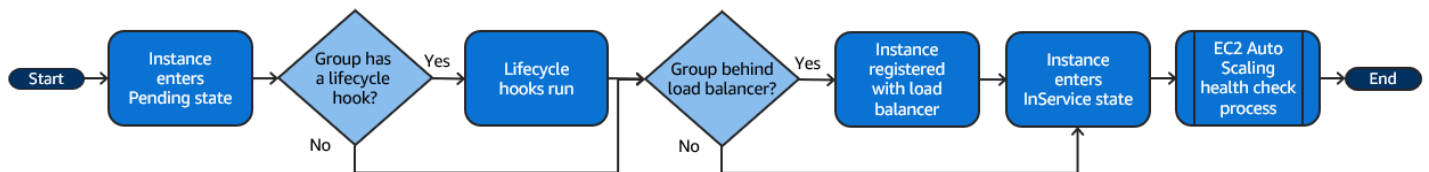
- Sie vergrößern die Gruppe manuell. Weitere Informationen finden Sie unter [Ändern der gewünschten Kapazität einer vorhandenen Auto-Scaling-Gruppe](#).
- Sie erstellen eine Skalierungsrichtlinie, mit der die Gruppe bei einem festgelegten Anstieg des Bedarfs automatisch vergrößert wird. Weitere Informationen finden Sie unter [Dynamische Skalierung für Amazon EC2 Auto Scaling](#).
- Sie richten die Skalierung nach Zeitplan ein, um die Gruppe zu einem bestimmten Zeitpunkt zu vergrößern. Weitere Informationen finden Sie unter [Geplante Skalierung für Amazon EC2 Auto Scaling](#).

Wenn ein Aufskalierungsereignis eintritt, startet die Auto-Scaling-Gruppe die erforderliche Anzahl von EC2-Instances unter Verwendung der ihr zugewiesenen Startvorlage. Zunächst lautet der Status der Instances Pending. Wenn Sie Ihrer Auto-Scaling-Gruppe einen Lebenszyklus-Hook hinzufügen, kann hier eine benutzerdefinierte Aktion ausgeführt werden. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks](#).

Wenn jede Instance vollständig konfiguriert ist und die Amazon EC2-Zustandsprüfungen besteht, wird sie der Auto-Scaling-Gruppe hinzugefügt und erhält den Zustand InService. Jede Instance wird auf die gewünschte Kapazität der Auto-Scaling-Gruppe angerechnet.

Wenn Ihre Auto-Scaling-Gruppe so konfiguriert ist, dass sie Datenverkehr von einem Load Balancer von Elastic Load Balancing empfängt, registriert Amazon EC2 Auto Scaling Ihre Instance automatisch beim Load Balancer, bevor es die Instance als InService markiert.

Im Folgenden werden die Schritte zur Registrierung einer Instance bei einem Load Balancer für ein Scale-Out-Event zusammengefasst.



In Betrieb genommene Instances

Instances behalten den Status InService, bis eines der folgenden Ereignisse eintritt:

- Zur horizontalen Skalierung nach unten beendet Amazon EC2 Auto Scaling diese Instance, um die Auto-Scaling-Gruppe zu verkleinern. Weitere Informationen finden Sie unter [Steuern welche Auto-Scaling-Instances beim Abskalieren beendet werden](#).
- Sie versetzen die Instance in den Status Standby. Weitere Informationen finden Sie unter [Aktivieren und Deaktivieren des Standby-Status](#).
- Sie trennen die Instance von der Auto-Scaling-Gruppe. Weitere Informationen finden Sie unter [Instances von Ihrer Auto Scaling Scaling-Gruppe trennen oder anhängen](#).
- Die Instance besteht eine bestimmte Anzahl an Zustandsprüfungen nicht und wird deshalb aus der Auto-Scaling-Gruppe entfernt, beendet und ersetzt. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Scale-In

Die folgenden Abwärtsskalierungsereignisse weisen die Auto-Scaling-Gruppe an, EC2-Instances von der Gruppe zu trennen und zu beenden:

- Sie verkleinern die Gruppe manuell. Weitere Informationen finden Sie unter [Ändern der gewünschten Kapazität einer vorhandenen Auto-Scaling-Gruppe](#).

- Sie erstellen eine Skalierungsrichtlinie, mit der die Gruppe bei einem festgelegten Rückgang des Bedarfs automatisch verkleinert wird. Weitere Informationen finden Sie unter [Dynamische Skalierung für Amazon EC2 Auto Scaling](#).
- Sie richten die Skalierung nach Zeitplan ein, um die Gruppe zu einem bestimmten Zeitpunkt zu verkleinern. Weitere Informationen finden Sie unter [Geplante Skalierung für Amazon EC2 Auto Scaling](#).

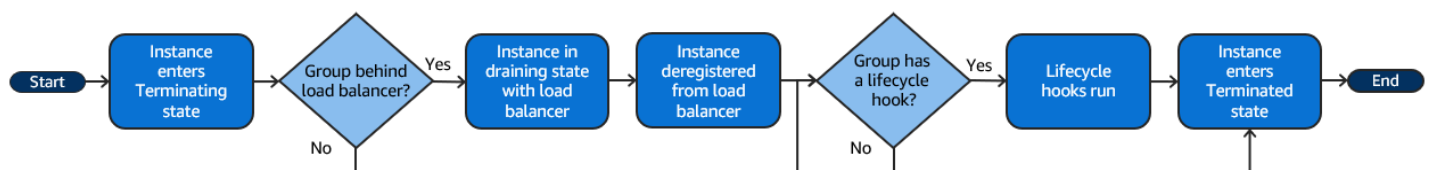
Es ist wichtig, dass Sie für jedes horizontale Skalierungsereignis ein entsprechendes Abwärtsskalierungsereignis erstellen. So stellen Sie sicher, dass die Ressourcen, die Ihrer Anwendung zugewiesen sind, dem Bedarf so gut wie möglich entsprechen.

Wenn ein Abwärtsskalierungsereignis auftritt, trennt die Auto-Scaling-Gruppe eine oder mehrere Instances ab. Die Auto-Scaling-Gruppe ermittelt anhand ihrer Beendigungsrichtlinie, welche Instance beendet werden soll. Instances, die gerade von der Auto-Scaling-Gruppe beendet werden, erhalten den Status `Terminating` und können nicht wieder in Betrieb genommen werden.

Wenn Ihre Auto-Scaling-Gruppe so konfiguriert ist, dass sie Datenverkehr von einem Load Balancer von Elastic Load Balancing empfängt, hebt Amazon EC2 Auto Scaling die Registrierung der beendenden Instance automatisch vom Load Balancer auf. Durch die Abmeldung der Instance wird sichergestellt, dass alle neuen Anforderungen an andere Instance in der Zielgruppe des Load Balancers umgeleitet werden, während bestehende Verbindungen mit der Instance fortgesetzt werden können, bis die Abmeldeverzögerung abläuft.

Wenn Sie Ihrer Auto-Scaling-Gruppe einen Lebenszyklus-Hook hinzufügen, kann eine benutzerdefinierte Aktion auf der Instance ausgeführt werden, die beendet wird. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks](#). Die Instance wird schließlich vollständig beendet und erhält den Status `Terminated`.

Im Folgenden werden die Schritte zum Abmelden einer Instance bei einem Load Balancer für ein Scale-In-Ereignis zusammengefasst.



Trennen einer Instance

Sie können eine Instance von Ihrer Auto-Scaling-Gruppe trennen. Nachdem die Instance getrennt wurde, können Sie sie getrennt von der Auto-Scaling-Gruppe verwalten oder sie an eine andere Auto-Scaling-Gruppe anfügen.

Weitere Informationen finden Sie unter [Instances von Ihrer Auto Scaling Scaling-Gruppe trennen oder anhängen](#).

Hinzufügen einer Instance

Sie können Ihrer Auto-Scaling-Gruppe eine laufende EC2-Instance hinzufügen, wenn sie bestimmte Kriterien erfüllt. Nach dem Hinzufügen der Instance wird sie als Teil der Auto-Scaling-Gruppe verwaltet.

Weitere Informationen finden Sie unter [Instances von Ihrer Auto Scaling Scaling-Gruppe trennen oder anhängen](#).

Lebenszyklus-Hooks

Sie können Ihrer Auto-Scaling-Gruppe einen Lebenszyklus-Hook hinzufügen, damit beim Starten oder Beenden von Instances benutzerdefinierte Aktionen ausgeführt werden.

Wenn Amazon-EC2-Auto-Scaling auf ein horizontales Skalierungsereignis reagiert, werden eine oder mehrere Instances gestartet. Zunächst lautet der Status der Instances Pending. Falls Sie Ihrer Auto-Scaling-Gruppe einen Lebenszyklus-Hook vom Typ `autoscaling:EC2_INSTANCE_LAUNCHING` hinzugefügt haben, ändert sich der Status der Instances von Pending zu `Pending:Wait`. Wenn Sie die Lebenszyklusaktion abgeschlossen haben, erhalten die Instances den Status `Pending:Proceed`. Wenn die Instances vollständig konfiguriert sind, werden sie der Auto-Scaling-Gruppe angefügt und erhalten den Status `InService`.

Wenn Amazon EC2 Auto Scaling auf ein Abwärtsskalierungsereignis reagiert, werden eine oder mehrere Instances beendet. Diese Instances werden von der Auto-Scaling-Gruppe getrennt und erhalten den Status `Terminating`. Falls Sie Ihrer Auto-Scaling-Gruppe einen Lebenszyklus-Hook vom Typ `autoscaling:EC2_INSTANCE_TERMINATING` hinzugefügt haben, ändert sich der Status der Instances von `Terminating` zu `Terminating:Wait`. Wenn Sie die Lebenszyklusaktion abgeschlossen haben, erhalten die Instances den Status `Terminating:Proceed`. Wenn die Instances vollständig beendet werden, erhalten sie den Status `Terminated`.

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).

Aktivieren und Deaktivieren des Standby-Status

Sie können eine Instance mit dem Status InService in den Status Standby versetzen. So können Sie die Instance aus dem Betrieb nehmen, Probleme beheben oder sie ändern und sie dann wieder in Betrieb nehmen.

Instances mit dem Status Standby werden weiterhin von der Auto-Scaling-Gruppe verwaltet. Sie werden jedoch erst wieder ein aktiver Teil Ihrer Anwendung, wenn Sie sie wieder in Betrieb nehmen.

Weitere Informationen finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#).

Kontingente für Auto Scaling Scaling-Ressourcen und Gruppen

Ihr AWS-Konto hat Standardkontingente, früher als Limits bezeichnet, für jeden AWS Dienst. Sofern nicht anders angegeben, gilt jedes Kontingent spezifisch für eine Region. Sie können Erhöhungen für einige Kontingente beantragen und andere Kontingente können nicht erhöht werden.

Um die Kontingente für Amazon EC2 Auto Scaling anzuzeigen, öffnen Sie die [Service-Quotas-Konsole](#). Wählen Sie im Navigationsbereich und dann AWS Services aus und wählen Sie Amazon EC2 Auto Scaling aus.

Informationen zum Beantragen einer Kontingenterhöhung finden Sie unter [Beantragen einer Kontingenterhöhung](#) im Service-Quotas-Benutzerhandbuch. Wenn das Kontingent unter Service Quotas noch nicht in verfügbar ist, verwenden Sie das [Formular Limits für Auto Scaling](#). Die Erhöhung eines Kontingents ist immer an die Region gebunden, für die sie angefragt wurde.

Amazon EC2 Auto Scaling-Ressourcen

Für Sie AWS-Konto gelten die folgenden Kontingente in Bezug auf die Anzahl der Auto Scaling Scaling-Gruppen und Startkonfigurationen, die Sie erstellen können.

Ressource	Standardkontingent
Auto-Scaling-Gruppen pro Region	500
Startkonfigurationen pro Region	200

Auto-Scaling-Gruppenkonfiguration

Ihr AWS-Konto hat die folgenden Kontingente in Bezug auf die Konfiguration von Auto Scaling Scaling-Gruppen. Sie können nicht geändert werden.

Ressource	Kontingent
Skalierungsrichtlinien pro Auto-Scaling-Gruppe	50
Geplante Vorgänge pro Auto-Scaling-Gruppe	125
Schrittanpassungen pro Richtlinie zur schrittweisen Skalierung	20
Lebenszyklus-Hooks pro Auto-Scaling-Gruppe	50
SNS-Themen pro Auto-Scaling-Gruppe	10
Classic Load Balancers pro Auto-Scaling-Gruppe	50
Elastic-Load-Balancing-Zielgruppen pro Auto-Scaling-Gruppe	50
VPC-Lattice-Zielgruppen pro Auto-Scaling-Gruppe	5

API-Operationen für Auto-Scaling-Gruppen

Amazon EC2 Auto Scaling bietet API-Vorgänge, mit denen Sie stapelweise Änderungen an Ihren Auto-Scaling-Gruppen vornehmen können. Im Folgenden sind die API-Grenzwerte für die maximale Anzahl von Elementen (maximale Array-Mitglieder) aufgeführt, die in einem einzigen Vorgang zulässig sind. Sie können nicht geändert werden.

Operation	Maximale Array-Mitglieder
AttachInstances	20-Instanz IDs
AttachLoadBalancers	10 Load Balancer
AttachLoadBalancerTargetGroups	10 Zielgruppen
BatchDeleteScheduledAction	50 geplante Aktionen

Operation	Maximale Array-Mitglieder
BatchPutScheduledUpdateGroupAction	50 geplante Aktionen
DetachInstances	20-Instanz IDs
DetachLoadBalancers	10 Load Balancer
DetachLoadBalancerTargetGroups	10 Zielgruppen
EnterStandby	20-Instanz IDs
ExitStandby	20-Instanz IDs
SetInstanceProtection	50 Instanz IDs

Drosselung von Anfragen für die Amazon EC2 Auto Scaling Scaling-API

Amazon EC2 Auto Scaling API-Anfragen werden mithilfe eines Token-Bucket-Schemas gedrosselt, um die Servicebandbreite aufrechtzuerhalten. Weitere Informationen finden Sie unter [API-Anforderungsrate](#) in der Amazon EC2 Auto Scaling API-Referenz.

EC2-Beendigungsraten

Amazon EC2 Auto Scaling bestimmt dynamisch die Anzahl der Vorgänge, die zu einem Zeitpunkt ausgeführt werden können, wenn Ihre Auto-Scaling-Gruppe abskaliert. Das bedeutet, dass die Anzahl der gleichzeitig beendeten Instances in den Auto-Scaling-Gruppen variieren kann. Diese Abweichungen werden durch externe Überlegungen verursacht, z. B. ob Amazon EC2 Auto Scaling Instances bei einem Load Balancer abmelden muss.

Sonstige -Services

Kontingente für andere Dienste wie Amazon EC2 und Amazon VPC können sich auf Ihre Auto Scaling Scaling-Gruppen auswirken. Sie können verwenden Service Quotas , um die Kontingente für EC2-Instances und andere Ressourcen in Ihrem zu aktualisieren. AWS-Konto In der Service Quotas Konsole können Sie alle Ihre verfügbaren Servicekontingenten einsehen und Erhöhungen für diese beantragen. Weitere Informationen finden Sie im Service Quotas -Benutzerhandbuch unter [Anfordern einer Kontingenterhöhung](#).

Informationen zu Kontingenten, die spezifisch für Startvorlagen sind, finden Sie unter [Einschränkungen für Startvorlagen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Für die Verwendung von Amazon EC2 Auto Scaling einrichten

Bevor Sie mit der Verwendung von Amazon EC2 Auto Scaling beginnen, führen Sie die folgenden Schritte durch.

Aufgaben

- [Bereiten Sie sich auf die Nutzung des vor AWS CLI](#)

Bereiten Sie sich auf die Nutzung des vor AWS CLI

Sie können die AWS Befehlszeilentools verwenden, um Befehle an der Befehlszeile Ihres Systems auszugeben, um Amazon EC2 Auto Scaling und andere AWS Aufgaben auszuführen.

Um die AWS Command Line Interface (AWS CLI) zu verwenden, laden Sie Version 1 oder 2 von herunter, installieren und konfigurieren Sie sie. AWS CLI Version 1 und 2 verfügen über die gleiche Amazon-EC2-Auto-Scaling-Funktionalität. Informationen zum Installieren der AWS CLI -Version 1 finden Sie unter [Installieren, Aktualisieren und Deinstallieren der AWS CLI](#) im AWS CLI -Version 1 Benutzerhandbuch. Informationen zur Installation der AWS CLI Version 2 finden Sie unter [Installation oder Aktualisierung der neuesten Version von AWS CLI](#) im AWS CLI Version 2-Benutzerhandbuch.

AWS CloudShell ermöglicht es Ihnen, die Installation AWS CLI in Ihrer Entwicklungsumgebung zu überspringen und sie AWS-Managementkonsole stattdessen in der zu verwenden. Sie vermeiden nicht nur die Installation, sondern müssen auch keine Anmeldeinformationen konfigurieren und keine Region angeben. Ihre AWS-Managementkonsole Sitzung bietet diesen Kontext für die AWS CLI. Sie können es verwenden AWS CloudShell , wenn es unterstützt wird AWS-Regionen. Weitere Informationen finden Sie unter [Erstellen Sie Auto Scaling Scaling-Gruppen über die Befehlszeile mit AWS CloudShell](#).

Weitere Informationen finden Sie unter [autoscaling](#) in der AWS CLI -Befehlsreferenz.

Erste Schritte mit Amazon EC2 Auto Scaling

Um mit Amazon EC2 Auto Scaling zu beginnen, können Sie Tutorials folgen, die Ihnen den Service vorstellen.

Themen

- [Tutorial: Erstellen Sie Ihre erste Auto Scaling Scaling-Gruppe](#)
- [Tutorial: Einrichten einer skalierten Anwendung mit Load Balancing](#)

Weitere Tutorials, die sich auf bestimmte Tools für die Verwaltung des Lebenszyklus von Instances in einer Auto Scaling Scaling-Gruppe konzentrieren, finden Sie in den folgenden Themen:

- [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#). Dieses Tutorial zeigt Ihnen, wie Sie Amazon verwenden, um Regeln EventBridge zu erstellen, die Lambda-Funktionen auf der Grundlage von Ereignissen aufrufen, die mit den Instances in Ihrer Auto Scaling Scaling-Gruppe passieren.
- [Tutorial: Verwenden Sie Datenskript- und Instance-Metadaten, um den Lebenszyklusstatus abzurufen](#). Dieses Tutorial zeigt Ihnen, wie Sie den Instance Metadata Service (IMDS) verwenden, um eine Aktion innerhalb der Instance selbst aufzurufen.

Bevor Sie eine Auto-Scaling-Gruppe für die Verwendung mit Ihrer Anwendung erstellen, prüfen Sie die Anwendung gründlich, während sie in der AWS Cloud ausgeführt wird. Berücksichtigen Sie dabei Folgendes:

- Wie viele Availability Zones soll die Auto-Scaling-Gruppe umfassen?
- Welche vorhandenen Ressourcen können verwendet werden, z. B. Sicherheitsgruppen oder Amazon Machine Images (AMIs).
- Egal ob Sie eine Skalierung durchführen möchten, um die Kapazität zu erhöhen oder zu verringern, oder ob Sie einfach nur sicherstellen möchten, dass immer eine bestimmte Anzahl von Servern läuft. Beachten Sie, dass Amazon EC2 Auto Scaling diese beiden Aufgaben gleichzeitig erfüllen kann.
- Welche Metriken sind für die Leistung Ihrer Anwendung am relevantesten?
- Wie lange es dauert, einen Server zu starten und bereitzustellen.

Je besser Sie Ihre Anwendung verstehen, desto effektiver können Sie die Auto Scaling-Architektur gestalten.

Tutorial: Erstellen Sie Ihre erste Auto Scaling Scaling-Gruppe

Dieses Tutorial bietet eine praktische Einführung in Amazon EC2 Auto Scaling über die AWS-Managementkonsole. Sie erstellen eine Startvorlage, die Ihre EC2-Instances und eine Auto Scaling Scaling-Gruppe mit einer einzigen Instance definiert. Nach dem Start Ihrer Auto Scaling Scaling-Gruppe beenden Sie die Instance und überprüfen, ob die Instance außer Betrieb genommen und ersetzt wurde. Um eine konstante Anzahl von Instances aufrechtzuerhalten, erkennt Amazon EC2 Auto Scaling automatisch die Zustands- und Erreichbarkeitsprüfungen von Amazon EC2 und reagiert darauf.

Wenn Sie sich für Amazon EC2 Auto Scaling anmelden, können Sie mit dem kostenlosen [Kontingent AWS kostenlos](#) loslegen. Sie können das kostenlose Kontingent verwenden, um eine `t2.micro`-Instance 12 Monate lang kostenlos zu starten und zu verwenden (in Regionen, in denen `t2.micro` nicht verfügbar ist, können Sie eine `t3.micro`-Instance im Rahmen des kostenlosen Kontingents verwenden). Wenn Sie eine Instance starten, die nicht vom kostenlosen Kontingent abgedeckt ist, fallen die standardmäßigen Amazon EC2-Nutzungsgebühren für die Instance an. Weitere Informationen finden Sie unter [Amazon EC2 – Preise](#).

Aufgaben

- [Vorbereitung auf den Walkthrough](#)
- [Schritt 1: Eine Startvorlage erstellen](#)
- [Schritt 2: Eine Auto-Scaling-Gruppe mit einer einzelnen Instance erstellen](#)
- [Schritt 3: Überprüfen Ihrer Auto-Scaling-Gruppe](#)
- [Schritt 4: Beenden einer Instance in Ihrer Auto-Scaling-Gruppe](#)
- [Schritt 5: Nächste Schritte](#)
- [Schritt 6: Bereinigen](#)

Vorbereitung auf den Walkthrough

In diesem Walkthrough wird davon ausgegangen, dass Sie sich mit dem Starten von EC2-Instances auskennen und bereits ein Schlüsselpaar und eine Sicherheitsgruppe erstellt haben.

Um mit der Verwendung von Amazon EC2 Auto Scaling zu beginnen, können Sie die Standard-VPC für Ihre verwenden. AWS-Konto Die Standard-VPC enthält ein öffentliches Standardsubnetz in jeder Availability Zone und ein Internet-Gateway, das Ihrer VPC zugeordnet ist. Sie können Ihre VPCs auf [Ihrer VPCs Seite](#) der Amazon Virtual Private Cloud (Amazon VPC) -Konsole einsehen.

Schritt 1: Eine Startvorlage erstellen

In diesem Schritt erstellen Sie eine Startvorlage, die den Typ der EC2-Instance angibt, die Amazon EC2 Auto Scaling für Sie erstellt. Nehmen Sie Informationen wie die ID des Amazon-Systemabbilds (Amazon Machine Image, AMI), den Instance-Typ, das Schlüsselpaar und die Sicherheitsgruppen auf.

Eine Startvorlage erstellen

1. Öffnen Sie die Amazon EC2 EC2-Konsole und rufen Sie die [Seite Launch Templates auf](#).
2. Wählen Sie in der oberen Navigationsleiste eine AWS-Region aus. Die Startvorlage und die Auto-Scaling-Gruppe, die Sie erstellen, sind an die von Ihnen angegebene Region gebunden.
3. Wählen Sie Startvorlage erstellen.
4. Geben Sie für Startvorlagenname **my-template-for-auto-scaling** ein.
5. Unter Auto-Scaling-Anleitung aktivieren Sie das Kontrollkästchen.
6. Wählen Sie für Application and OS Images (Amazon Machine Image) (Anwendungs- und Betriebssystem-Images (Amazon Machine Image)) eine Version von Amazon Linux 2 (HVM) aus der Liste Quick Start (Schnellstart) aus. Das AMI dient als grundlegende Konfigurationsvorlage für Ihre Instances.
7. Wählen Sie für Instance type (Instance-Typ) eine Hardwarekonfiguration aus, die mit dem von Ihnen angegebenen AMI kompatibel ist.
8. (Optional) Wählen Sie für Key pair name (Schlüsselpaarnamen) ein vorhandenes Schlüsselpaar aus. Sie verwenden Schlüsselpaare, um eine Verbindung zu einer Amazon-EC2-Instance mit SSH herzustellen. Informationen dazu, wie eine Verbindung mit einer Instance herstellen, sind nicht Bestandteil dieses Tutorials. Daher müssen Sie kein Schlüsselpaar angeben, es sei denn, Sie möchten eine Verbindung mit der Instance über SSH herstellen.
9. Für Network settings (Netzwerkeinstellungen) erweitern Sie die Option Advanced network configuration (Erweiterte Netzwerkkonfiguration) und tun Folgendes:
 - a. Wählen Sie zum Konfigurieren der primären Netzwerkschnittstelle Add network interface (Netzwerkschnittstelle hinzufügen) aus.

- b. Geben Sie für Auto-Assign Public IP an, ob Ihre Instance eine öffentliche IPv4-Adresse erhält. Standardmäßig weist Amazon EC2 eine öffentliche IPv4-Adresse zu, wenn die EC2-Instance in einem Standardsubnetz gestartet wird oder wenn die Instance in einem Subnetz gestartet wird, das für die automatische Zuweisung einer öffentlichen Adresse konfiguriert wurde. IPv4 Wenn Sie keine Verbindung zu Ihrer Instance herstellen müssen, wählen Sie Disable.
 - c. Wählen Sie als Sicherheitsgruppen-ID eine Sicherheitsgruppe in derselben VPC aus, die Sie als VPC für Ihre Auto Scaling Scaling-Gruppe verwenden möchten. Wenn Sie keine Sicherheitsgruppe angeben, wird die Instance automatisch der Standard-Sicherheitsgruppe für die VPC zugeordnet.
 - d. Wählen Sie für Bei Kündigung löschen die Option Ja aus, um die Netzwerkschnittstelle zu löschen, wenn die Instance gelöscht wird.
10. Wählen Sie Startvorlage erstellen.
 11. Wählen Sie auf der Bestätigungsseite Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Schritt 2: Eine Auto-Scaling-Gruppe mit einer einzelnen Instance erstellen

Gehen Sie wie folgt vor, um dort fortzufahren, wo Sie nach der Erstellung einer Startvorlage aufgehört haben.

So erstellen Sie eine Auto Scaling-Gruppe

1. Geben Sie auf der Seite Choose launch template or configuration (Startvorlage oder Konfiguration auswählen) als Name der Auto-Scaling-Gruppen **my-first-asg** ein.
2. Wählen Sie Weiter aus.

Die Seite „Instance-Startoptionen auswählen“ wird angezeigt, auf der Sie die VPC-Netzwerkeinstellungen auswählen können, die die Auto Scaling Scaling-Gruppe verwenden soll, und die Ihnen Optionen für den Start von On-Demand- und Spot-Instances bietet.

3. Lassen Sie VPC im Bereich Netzwerk auf die von Ihnen gewählte Standard-VPC eingestellt AWS-Region, oder wählen Sie Ihre eigene VPC aus. Die Standard-VPC wird automatisch so konfiguriert, dass sie eine Internetverbindung für Ihre Instance bereitstellt. Diese VPC umfasst ein öffentliches Subnetz in jeder Availability Zone in der Region.
4. Wählen Sie für Availability Zones and subnets (Subnetze) ein Subnetz für jede Availability Zone aus, die Sie einschließen möchten. Verwenden Sie Subnetze in mehreren Availability Zones, um

eine hohe Verfügbarkeit zu erzielen. Weitere Informationen finden Sie unter [Überlegungen bei der Auswahl von VPC-Subnetzen](#).

5. Verwenden Sie im Abschnitt Instance type requirements (Anforderungen an den Instance-Typ) die Standardeinstellung, um diesen Schritt zu vereinfachen. (Setzen Sie die Startvorlage nicht außer Kraft.) In diesem Tutorial werden Sie nur eine On-Demand-Instance mit dem in Ihrer Startvorlage angegebenen Instance-Typ starten.
6. Behalten Sie die restlichen Standardeinstellungen für dieses Tutorial bei, und wählen Sie Skip to review (Mit Prüfen fortfahren).

Note

Die anfängliche Größe der Gruppe wird durch ihre gewünschte Kapazität bestimmt. Der Standardwert ist 1-Instance.

7. Überprüfen Sie auf der Seite Review (Überprüfen) die Informationen für die Gruppe und wählen Sie dann Create Auto Scaling Group (Auto-Scaling-Gruppe erstellen) aus.

Schritt 3: Überprüfen Ihrer Auto-Scaling-Gruppe

Nun, da Sie eine Auto-Scaling-Gruppe erstellt haben, können Sie überprüfen, ob die Gruppe eine EC2-Instance gestartet hat.

Tip

Im folgenden Verfahren sehen Sie sich die Abschnitte Activity history (Verlauf der Aktivität) und Instances für die Auto-Scaling-Gruppe an. In beiden sollten die benannten Spalten bereits angezeigt werden. Um ausgeblendete Spalten anzuzeigen oder die Anzahl der angezeigten Zeilen zu ändern, wählen Sie das Zahnradsymbol in der oberen rechten Ecke jedes Abschnitts, um die Einstellungen zu öffnen, die Einstellungen nach Bedarf zu aktualisieren und Confirm (Bestätigen) auszuwählen.

So prüfen Sie, dass Ihre Auto-Scaling-Gruppe eine EC2-Instance gestartet hat

1. Öffnen Sie die Seite [Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe, die Sie gerade erstellt haben.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet. Die erste verfügbare Registerkarte ist die Registerkarte Details, die Informationen zur Auto-Scaling-Gruppe anzeigt.

3. Wählen Sie die zweite Registerkarte mit der Bezeichnung Activity (Aktivität). Unter Activity history (Aktivitätsverlauf) können Sie sich den Fortschritt der Aktivitäten anzeigen lassen, die der Auto-Scaling-Gruppe zugeordnet sind. In der Status-Spalte wird der aktuelle Status Ihrer Instance angezeigt. Während die Instance gestartet wird, zeigt die Statusspalte `Not yet in service` an. Nach dem Start der Instance ändert sich der Status in `Successful`. Sie können auch die Aktualisierungsschaltfläche verwenden, um den aktuellen Status der Instance anzuzeigen.
4. Auf der Registerkarte Instance management (Instance-Verwaltung), unter Instances, können Sie sich den Status der Instance ansehen.
5. Stellen Sie sicher, dass Ihre Instance erfolgreich gestartet wurde. Es dauert einige Zeit, bis die Instance startet.
 - In der Spalte Lifecycle (Lebenszyklus) wird Ihnen der Zustand Ihrer Instance angezeigt. Die Instance befindet sich zunächst im Status `Pending`. Wenn eine Instance für den Empfang von Datenverkehr bereit ist, lautet der Status `InService`.
 - In der Spalte Health Status wird das Ergebnis der Amazon EC2 Auto Scaling Scaling-Zustandsprüfungen für Ihre Instance angezeigt.

Schritt 4: Beenden einer Instance in Ihrer Auto-Scaling-Gruppe

Sie können diese Schritte nutzen, um mehr über die Funktionsweise von Amazon EC2 Auto Scaling zu erfahren, insbesondere darüber, wie es bei Bedarf neue Instances startet. Die Mindestgröße für die Auto-Scaling-Gruppe, die Sie in diesem Tutorial erstellt haben, ist eine Instance. Deshalb muss Amazon EC2 Auto Scaling eine neue Instance als Ersatz starten, wenn Sie diese laufende Instance beenden.

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.
3. Wählen Sie auf der Registerkarte Instance management (Instance-Verwaltung) unter Instances die ID der Instance aus.

Dadurch gelangen Sie zur Seite Instances in der Amazon-EC2-Konsole, auf der Sie die Instance beenden können.

4. Wählen Sie Actions, Instance State und Terminate aus. Wählen Sie Yes, Terminate aus, wenn Sie zum Bestätigen aufgefordert werden.
5. Wählen Sie im Navigationsbereich unter Auto Scaling Auto Scaling Groups (Auto Scaling-Gruppe) aus. Wählen Sie Ihre Auto-Scaling-Gruppe aus und wählen Sie die Registerkarte Activity (Aktivität).

Wenn Sie eine Instance auf der Instance-Seite beenden, dauert es nach dem Beenden der Instance ein oder zwei Minuten, bis eine neue Instance gestartet wird. Im Aktivitätsverlauf sehen Sie beim Start der Skalierungsaktivität einen Eintrag für die Beendigung der ersten Instance und einen Eintrag für den Start einer neuen Instance. Verwenden Sie die Schaltfläche „Aktualisieren“, bis Sie die neuen Einträge sehen.

6. Auf der Registerkarte Instance management (Instance-Verwaltung) wird im Abschnitt Instances nur die neue Instance angezeigt.
7. Wählen Sie im Navigationsbereich unter Instances die Option Instances aus. Diese Seite zeigt sowohl die beendete als auch die neue laufende Instance.

Schritt 5: Nächste Schritte

Fahren Sie mit dem nächsten Schritt fort, wenn Sie die Basisinfrastruktur löschen möchten, die Sie gerade erstellt haben. Andernfalls können Sie diese Infrastruktur als Grundlage verwenden und die folgenden Aktionen ausprobieren:

- Herstellen einer Verbindung zu Ihrer Linux-Instance über Session Manager. Weitere Informationen finden Sie unter [Connect zu Ihrer EC2-Instance mithilfe von Session Manager](#) und [Connect zu Ihrer Linux-Instance mithilfe von SSH](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Konfigurieren Sie eine SNS-Benachrichtigung, die Sie benachrichtigt, sobald Ihre Auto-Scaling-Gruppe Instances startet oder beendet. Weitere Informationen finden Sie unter [Amazon SNS SNS-Benachrichtigungsoptionen](#).
- Skalieren Sie Ihre Auto-Scaling-Gruppe manuell, um die SNS-Benachrichtigung zu testen. Weitere Informationen finden Sie unter [Ändern der gewünschten Kapazität einer Auto-Scaling-Gruppe](#).

Sie können sich auch mit den Konzepten der auto Skalierung vertraut machen, indem Sie [Skalierungsrichtlinien für die Ziel-Nachverfolgung](#). Wenn sich die Auslastung Ihrer Anwendung ändert, kann Ihre Auto Scaling-Gruppe automatisch aufskalieren (Instanzen hinzufügen) und abskalieren (weniger Instanzen ausführen), indem die gewünschte Kapazität der Gruppe zwischen

der minimalen und der maximalen Kapazitätsgrenze angepasst wird. Weitere Informationen zu diesen Limits finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).

Schritt 6: Bereinigen

Sie können entweder Ihre Skalierungsinfrastruktur löschen oder nur Ihre Auto Scaling Scaling-Gruppe löschen und Ihre Startvorlage behalten, um sie später zu verwenden.

Wenn Sie eine Instance gestartet haben, die nicht unter das [kostenlose Kontingent für AWS](#) fällt, sollten Sie die Instance beenden, um zusätzliche Gebühren zu vermeiden. Wenn Sie die Instance beenden, werden auch die damit verknüpften Daten gelöscht.

So löschen Sie Ihre Auto-Scaling-Gruppe

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe (my-first-asg).
3. Wählen Sie Löschen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **delete** ein, um das Löschen der angegebenen Auto-Scaling-Gruppe zu löschen, wählen Sie dann Löschen.

Ein Ladesymbol in der Spalte Name zeigt an, dass die Auto-Scaling-Gruppe gelöscht wird. Wenn der Löschvorgang erfolgt ist, zeigen die Spalten Desired (Gewünscht), Min und Max 0-Instances für die Auto-Scaling-Gruppe an. Es dauert einige Minuten, bis die Instance beendet und die Gruppe gelöscht werden. Aktualisieren Sie die Liste, um den aktuellen Status anzuzeigen.

Überspringen Sie folgenden Schritte, falls Sie die Startvorlage behalten möchten.

So löschen Sie eine Startvorlage

1. Öffnen Sie die Seite [Startvorlagen](#) der Amazon EC2 Konsole.
2. Wählen Sie Ihre Startvorlage aus (my-template-for-auto-scaling).
3. Wählen Sie Actions (Aktionen) und Delete template (Vorlage löschen) aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **Delete** ein, um das Löschen der angegebenen Auto-Scaling-Gruppe zu bestätigen, wählen Sie dann Löschen.

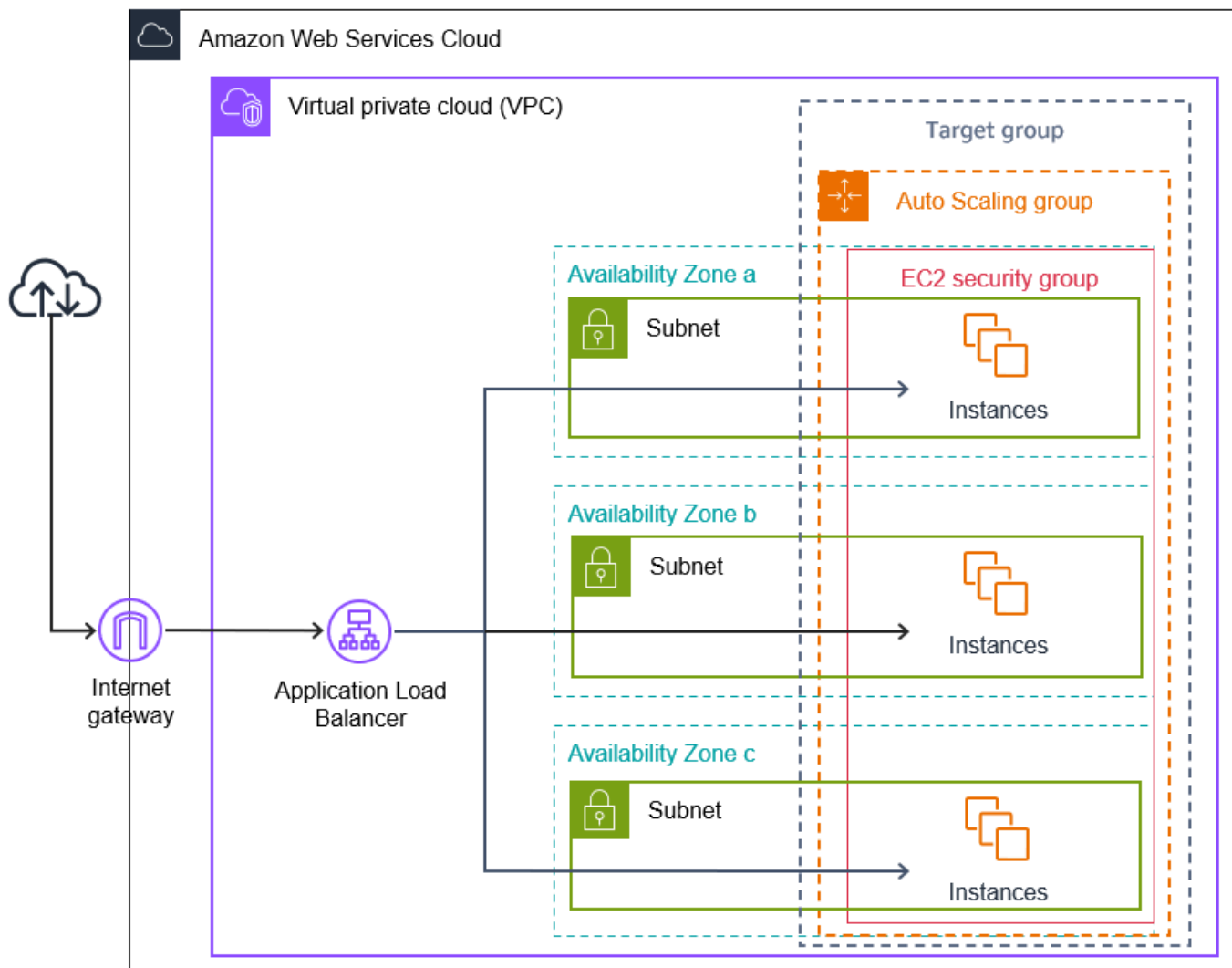
Tutorial: Einrichten einer skalierten Anwendung mit Load Balancing

Important

Bevor Sie sich mit diesem Tutorial befassen, empfehlen wir Ihnen, zunächst das folgende einführende Tutorial zu lesen: [Erstellen Sie Ihre erste Auto Scaling Scaling-Gruppe](#).

Wenn Sie Ihre Auto-Scaling-Gruppe bei einem Elastic Load Balancing Load Balancer registrieren, können Sie eine Anwendung mit Lastenausgleich einrichten. Elastic Load Balancing arbeitet mit Amazon EC2 Auto Scaling zusammen, um eingehenden Datenverkehr auf Ihre gesunden Amazon EC2-Instances zu verteilen. Dies erhöht die Skalierbarkeit und Verfügbarkeit Ihrer Anwendung. Sie können Elastic Load Balancing innerhalb mehrerer Availability Zones aktivieren, um die Fehlertoleranz Ihrer Anwendungen zu erhöhen.

In diesem Tutorial werden die grundlegenden Schritte zum Einrichten einer Anwendung mit Lastenausgleich beschrieben, wenn die Auto-Scaling-Gruppe erstellt wird. Wenn Sie fertig sind, sollte Ihre Architektur wie das folgende Diagramm aussehen:



Elastic Load Balancing unterstützt verschiedene Load Balancer-Typen. Wir empfehlen Ihnen, für diese praktische Anleitung einen Application Load Balancer zu verwenden.

Weitere Informationen zum Einführen eines Load Balancer in Ihre Architektur finden Sie unter [Verwenden Sie Elastic Load Balancing, um den eingehenden Anwendungsdatenverkehr in Ihrer Auto Scaling Scaling-Gruppe zu verteilen.](#)

Aufgaben

- [Voraussetzungen](#)
- [Schritt 1: Einrichten einer Startvorlage oder Startkonfiguration](#)
- [Schritt 2: Erstellen einer Auto-Scaling-Gruppe](#)
- [Schritt 3: Überprüfen Sie, ob Ihr Load Balancer angefügt ist](#)
- [Schritt 4: Nächste Schritte](#)

- [Schritt 5: Bereinigen](#)
- [Zugehörige Ressourcen](#)

Voraussetzungen

- Einen Load Balancer und eine Zielgruppe. Stellen Sie sicher, dass Sie die gleiche Availability Zones für den Load Balancer auswählen, die Sie auch für Ihre Auto-Scaling-Gruppe aktivieren möchten. Weitere Informationen finden Sie unter [Erste Schritte mit Elastic Load Balancing](#) im Elastic Load Balancing-Benutzerhandbuch.
- Eine Sicherheitsgruppe für Ihre Startvorlage oder Startkonfiguration. Die Sicherheitsgruppe muss den Zugriff vom Load Balancer sowohl auf den Listener-Port (normalerweise Port 80 für HTTP-Datenverkehr) als auch auf den Port, den Elastic Load Balancing für Zustandsprüfungen verwenden soll, erlauben. Weitere Informationen finden Sie in der entsprechenden Dokumentation:
 - [Zielsicherheitsgruppen](#) im Benutzerhandbuch für Application Load Balancer
 - [Zielsicherheitsgruppen](#) im Benutzerhandbuch für Network Load Balancer

Wenn die Instances öffentliche IP-Adressen aufweisen, können Sie optional SSH-Datenverkehr zulassen, wenn Sie eine Verbindung zu den Instances herstellen müssen.

- (Optional) Eine IAM-Rolle, die Ihrer Anwendung Zugriff auf AWS gewährt.
- (Optional) Sie haben ein Amazon Machine Image (AMI) als Quellvorlage für Ihre Amazon EC2-Instances definiert. Um jetzt eine zu erstellen, starten Sie eine Instance. Geben Sie die IAM-Rolle (sofern erstellt) und alle benötigten Konfigurationsskripts als Benutzerdaten an. Stellen Sie eine Verbindung mit der Instance her und passen Sie sie an. Beispielsweise können Sie Software und Anwendungen installieren, Daten kopieren und weitere EBS-Volumes anfügen. Testen Sie Ihre Anwendungen auf der Instance, um sicherzustellen, dass sie ordnungsgemäß konfiguriert ist. Speichern Sie diese aktualisierte Konfiguration als benutzerdefiniertes AMI. Sie können die Instance beenden, wenn Sie sie zu einem späteren Zeitpunkt nicht mehr benötigen. Instances, die über dieses neue benutzerdefinierte AMI gestartet werden, enthalten die Anpassungen, die Sie beim Erstellen des AMI vorgenommen haben.
- Eine Virtual Private Cloud (VPC). Dieses Tutorial bezieht sich auf die Standard-VPC, Sie können aber auch Ihre eigene verwenden. Stellen Sie in letzterem Fall sicher, dass Ihre VPC über ein Subnetz verfügt, das jeder Availability Zone der Region zugeordnet ist, in der Sie arbeiten. Mindestens zwei öffentliche Subnetze müssen verfügbar sein, um den Load Balancer zu erstellen. Sie müssen außerdem über zwei private Subnetze oder zwei öffentliche Subnetze verfügen, um Ihre Auto-Scaling-Gruppe zu erstellen und sie beim Load Balancer zu registrieren.

Schritt 1: Einrichten einer Startvorlage oder Startkonfiguration

Verwenden Sie entweder eine Startvorlage oder eine Startkonfiguration für diese praktische Anleitung.

Themen

- [Wählen oder erstellen Sie eine Startvorlage](#)
- [Auswählen oder Erstellen einer Startkonfiguration](#)

Wählen oder erstellen Sie eine Startvorlage

Wenn Sie bereits über eine Startvorlage verfügen, die Sie verwenden möchten, wählen Sie sie wie folgt aus.

So wählen Sie eine vorhandene Startvorlage aus

1. Öffnen Sie die Seite [Startvorlagen](#) der Amazon EC2 Konsole.
2. Wählen Sie in der Navigationsleiste am oberen Bildschirmrand die Region aus, in der der Load Balancer erstellt wurde.
3. Wählen Sie eine Startvorlage aus.
4. Wählen Sie Actions (Aktionen), Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Alternativ können Sie wie folgt eine neue Startvorlage erstellen.

Eine Startvorlage erstellen

1. Öffnen Sie die Seite [Startvorlagen](#) der Amazon EC2 Konsole.
2. Wählen Sie in der Navigationsleiste am oberen Bildschirmrand die Region aus, in der der Load Balancer erstellt wurde.
3. Wählen Sie Startvorlage erstellen.
4. Geben Sie einen Namen und eine Beschreibung für die anfängliche Version der Startvorlage ein.
5. Wählen Sie für Application and OS Images (Amazon Machine Image) (Anwendungs- und Betriebssystem-Images (Amazon Machine Image)) die ID der AMI für Ihre Instances aus. Sie können alle verfügbaren AMIs durchsuchen oder ein AMI aus der Liste „Zuletzt verwendet“ oder „Schnellstart“ auswählen. Wenn Sie das benötigte AMI nicht sehen, wählen Sie Browse more, AMIs um den vollständigen AMI-Katalog zu durchsuchen.

6. Wählen Sie für Instance type eine Hardwarekonfiguration für Ihre Instances aus, die mit dem von Ihnen angegebenen AMI kompatibel ist.
7. (Optional) Geben Sie für Schlüsselpaar (Anmeldung) das Schlüsselpaar ein, das Sie bei der Verbindung mit Ihren Instances verwenden möchten.
8. Für Network settings (Netzwerkeinstellungen) erweitern Sie die Option Advanced network configuration (Erweiterte Netzwerkkonfiguration) und tun Folgendes:
 - a. Wählen Sie zum Konfigurieren der primären Netzwerkschnittstelle Add network interface (Netzwerkschnittstelle hinzufügen) aus.
 - b. Geben Sie für Auto-Assign Public IP an, ob Ihre Instances öffentliche IPv4-Adressen erhalten. Standardmäßig weist Amazon EC2 eine öffentliche IPv4-Adresse zu, wenn die EC2-Instance in einem Standardsubnetz gestartet wird oder wenn die Instance in einem Subnetz gestartet wird, das für die automatische Zuweisung einer öffentlichen Adresse konfiguriert wurde. IPv4 Wenn Sie keine Verbindung zu Ihren Instances herstellen müssen, können Sie Disable wählen, um zu verhindern, dass Instances in Ihrer Gruppe Traffic direkt aus dem Internet empfangen. In diesem Fall empfangen sie nur den Datenverkehr vom Load Balancer.
 - c. Für Sicherheitsgruppen-ID geben Sie eine Sicherheitsgruppe für Ihre Instances aus derselben VPC wie dem Load Balancer an.
 - d. Für Beim Beenden löschen wählen Sie Ja aus. Damit wird die Netzwerkschnittstelle gelöscht, wenn die Auto-Scaling-Gruppe nach unten skaliert wird und die Instance, der die Netzwerkschnittstelle angefügt ist, beendet wird.
9. (Optional) Um Anmeldeinformationen für Ihre Instances sicher zu verteilen, geben Sie bei Advanced details (Erweiterte Details), IAM instance profile (IAM-Instance-Profil) den Amazon-Ressourcennamen (ARN) Ihrer IAM-Rolle ein.
10. (Optional) Wenn Sie Benutzerdaten oder ein Konfigurationsskript für Ihre Instances angeben möchten, fügen Sie dies unter Advanced Details, User data ein.
11. Wählen Sie Startvorlage erstellen.
12. Wählen Sie auf der Bestätigungsseite Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Auswählen oder Erstellen einer Startkonfiguration

Note

Wir raten dringend davon ab, Startkonfigurationen in neuen Anwendungen zu verwenden, da es sich um eine veraltete Funktion handelt, für die keine geplanten Investitionen erforderlich sind. Darüber hinaus haben neue Konten, die am oder nach dem 1. Juni 2023 erstellt wurden, nicht die Möglichkeit, neue Startkonfigurationen über die Konsole zu erstellen. Weitere Informationen finden Sie unter [Auto Scaling Scaling-Startkonfigurationen](#).

So wählen Sie eine vorhandene Startkonfiguration aus

1. Öffnen Sie die Seite [Konfigurationen starten](#) der Amazon EC2-Konsole.
2. Wählen Sie in der oberen Navigationsleiste die Region aus, in der der Load Balancer erstellt wurde.
3. Wählen Sie eine Startkonfiguration aus.
4. Wählen Sie Actions (Aktionen), Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Alternativ können Sie wie folgt eine neue Startkonfiguration erstellen.

Erstellen Sie eine Startkonfiguration wie folgt:

1. Öffnen Sie die Seite [Konfigurationen starten](#) der Amazon EC2-Konsole. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Startkonfigurationen anzeigen aus, um zu bestätigen, dass Sie die Seite Startkonfigurationen aufrufen möchten.
2. Wählen Sie in der oberen Navigationsleiste die Region aus, in der der Load Balancer erstellt wurde.
3. Klicken Sie auf Erstellen einer Startkonfiguration und geben Sie einen Namen für die Startkonfiguration ein.
4. Geben Sie für Amazon Machine Image (AMI), die ID des AMI für Ihre Instances als Suchkriterien ein.
5. Für Instance-Typ wählen Sie eine Hardware-Konfiguration für Ihre Instance aus.
6. Unter Zusätzliche Konfiguration achten Sie auf die folgenden Felder:

- a. (Optional) Wählen Sie zum sicheren Verteilen von Anmeldeinformationen an die EC2-Instance für IAM-Instance-Profil Ihre IAM-Rolle aus. Weitere Informationen finden Sie unter [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#).
 - b. (Optional) Wenn Sie Benutzerdaten oder ein Konfigurationsskript für Ihre Instance angeben möchten, fügen Sie dies unter Advanced Details, User data ein.
 - c. (Optional) Behalten Sie für Erweiterte Details, IP-Adresstyp den Standardwert bei. Wenn Sie Ihre Auto-Scaling-Gruppe erstellen, können Sie Instances in Ihrer Auto-Scaling-Gruppe eine öffentliche IP-Adresse zuweisen, indem Sie Subnetze verwenden, für die das Attribut der öffentlichen IP-Adressierung aktiviert ist, z. B. die Standard-Subnetze in der Standard-VPC. Wenn Sie keine Verbindung zu Ihren Instances herstellen müssen, können Sie auch Keine öffentliche IP-Adresse Instances hinzuweisen auswählen, um zu verhindern, dass Instances in Ihrer Gruppe Datenverkehr direkt aus dem Internet empfangen. In diesem Fall empfangen sie nur den Datenverkehr vom Load Balancer.
7. Wählen Sie für Sicherheitsgruppen eine vorhandene Sicherheitsgruppe aus derselben VPC wie dem Load Balancer aus. Wenn Sie die Option Erstellen einer neuen Sicherheitsgruppe ausgewählt lassen, wird eine Standard-SSH-Regel für Amazon-EC2-Instances konfiguriert, auf denen Linux ausgeführt wird. Für Amazon-EC2-Instances, die Windows ausführen, wird eine Standard-RDP-Rolle konfiguriert.
 8. Für Schlüsselpaar (Login) wählen Sie eine Option unter Optionen für Schlüsselpaar aus.

Wenn Sie ein Amazon EC2-Instance-Schlüsselpaar bereits konfiguriert haben, können Sie das Schlüsselpaar hier auswählen.

Wenn Sie noch kein Amazon EC2-Instance-Schlüsselpaar haben, klicken Sie auf Create a new key pair (Ein neues Schlüsselpaar erstellen) und geben Sie einen wiedererkennbaren Namen ein. Wählen Sie Download Key Pair (Schlüsselpaar herunterladen) aus, um das Schlüsselpaar auf Ihrem Computer herunterzuladen.

 **Important**

Wählen Sie nicht Proceed without a key pair (Ohne Schlüsselpaar fortfahren) aus, wenn Sie eine Verbindung mit Ihrer Instance herstellen müssen.

9. Aktivieren Sie das Bestätigungskontrollkästchen und wählen Sie dann Create launch configuration (Startkonfiguration erstellen) aus.

10. Aktivieren Sie das Kontrollkästchen neben dem Namen der neuen Startkonfiguration und wählen Sie Aktionen, Auto-Scaling-Gruppe erstellen aus.

Schritt 2: Erstellen einer Auto-Scaling-Gruppe

Gehen Sie wie folgt vor, um den Vorgang dort fortzusetzen, wo Sie ihn nach dem Erstellen oder Auswählen der Startvorlage oder der Startkonfiguration abgebrochen haben.

So erstellen Sie eine Auto-Scaling-Gruppe

1. Geben Sie auf der Seite Startvorlage oder -konfiguration auswählen für Auto-Scaling-Gruppenname einen Namen für Ihre Auto-Scaling-Gruppe ein.
2. [Nur Startvorlage] Wählen Sie unter Launch template version (Version der Startvorlage) aus, ob die Auto Scaling-Gruppe beim horizontalen Skalieren nach oben die standardmäßige, die neueste oder eine bestimmte Version der Startvorlage verwenden soll.
3. Wählen Sie Weiter aus.

Die Seite Choose instance launch options (Startoptionen für Instances auswählen) wird angezeigt. Hier können Sie die VPC-Netzwerkeinstellungen auswählen, welche die Auto-Scaling-Gruppe verwenden soll, und Sie erhalten Optionen für den Start von On-Demand- und Spot-Instances (wenn Sie eine Startvorlage ausgewählt haben).

4. Wählen Sie im Abschnitt Netzwerk für VPC die VPC, die Sie für Ihren Load Balancer verwendet haben. Wenn Sie die Standard-VPC auswählen, wird sie automatisch so konfiguriert, dass sie Internetkonnektivität für Ihre Instances bereitstellt. Diese VPC umfasst ein öffentliches Subnetz in jeder Availability Zone in der Region.
5. Wählen Sie für Availability Zone and Subnets (Subnetze) mindestens ein Subnetz aus jeder Availability Zone aus, das Sie einbeziehen möchten, basierend auf den Availability Zones, in denen sich der Load Balancer befindet. Weitere Informationen finden Sie unter [Überlegungen bei der Auswahl von VPC-Subnetzen](#).
6. [Nur Vorlage starten] Im Abschnitt Anforderungen an den Instance-Typ verwenden Sie die Standardeinstellung, um diesen Schritt zu vereinfachen. (Setzen Sie die Startvorlage nicht außer Kraft.) In diesem Tutorial werden Sie nur On-Demand-Instances mit dem in Ihrer Startvorlage angegebenen Instance-Typ starten.
7. Wählen Sie Next (Weiter), um zur Seite Configure advanced options (Erweiterte Optionen konfigurieren) zu gelangen.

8. Um die Gruppe an einen bestehenden Load-Balancer anzuhängen, wählen Sie im Abschnitt Load balancing (Lastenverteilung vornehmen) die Option Attach to an existing load balancer (An eine bestehende Lastenverteilung anhängen). Sie können Wählen Sie aus Ihren Load-Balancer-Zielgruppen oder Wählen Sie aus den klassischen Load Balancern auswählen. Sie können dann den Namen einer Zielgruppe für den von Ihnen erstellten Application Load Balancer oder Network Load Balancer wählen oder den Namen eines Classic Load Balancers auswählen.
9. (Optional) Wählen Sie für Zustandsprüfungen und Zusätzliche Zustandsprüfungstypen die Option Elastic Load Balancing-Zustandsprüfungen aktivieren aus.
10. (Optional) Geben Sie unter Karenzzeit für die Zustandsprüfung die Zeit in Sekunden ein. Diese Zeitspanne gibt an, wie lange Amazon EC2 Auto Scaling warten muss, bevor der Integritätsstatus einer Instance überprüft wird, nachdem sie den InService Status erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).
11. Wenn Sie die Konfiguration der Auto-Scaling-Gruppe abgeschlossen haben, wählen Sie Skip to review (Mit Prüfung fortfahren) aus.
12. Prüfen Sie auf der Seite Review (Überprüfung) die Details Ihrer Auto-Scaling-Gruppe. Sie können Edit (Bearbeiten) auswählen, um Änderungen vorzunehmen. Wählen Sie Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus, wenn Sie fertig sind.

Nachdem Sie die Auto-Scaling-Gruppe mit dem angefügtem Load Balancer erstellt haben, registriert der Load Balancer automatisch neue Instances, sobald diese online geschaltet werden. Sie haben zu diesem Zeitpunkt nur eine Instance, daher gibt es nicht viel zu registrieren. Sie können jedoch zusätzliche Instances hinzufügen, indem Sie die gewünschte Kapazität der Gruppe aktualisieren. step-by-stepAnweisungen finden Sie unter [Ändern der gewünschten Kapazität einer Auto-Scaling-Gruppe](#).

Schritt 3: Überprüfen Sie, ob Ihr Load Balancer angefügt ist

So überprüfen Sie, ob Ihr Load Balancer angefügt ist

1. Wählen Sie auf der Seite [Auto-Scaling-Gruppen](#) Amazon EC2-Konsole das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe aus.
2. Auf der Registerkarte Details werden unter Load Balancing (Lastenausgleich) alle angefügten Load Balancer-Zielgruppen oder Classic Load Balancers angezeigt.
3. Auf der Registerkarte Aktivität können Sie unter Aktivitätsverlauf überprüfen, ob Ihre Instances erfolgreich gestartet wurden. Die Spalte Status zeigt an, ob Ihre Auto-Scaling-Gruppe erfolgreich

Instances gestartet hat. Wenn Ihre Instances nicht gestartet werden können, finden Sie Tipps zur Fehlerbehebung für häufige Instance-Startprobleme unter [Probleme in Amazon EC2 Auto Scaling beheben](#).

4. Sie können auf der Registerkarte Instance-Verwaltung unter Instances überprüfen, ob Ihre Instances Verkehr empfangen können. Anfänglich sind Ihre Instances im Status Pending. Wenn eine Instance für den Empfang von Datenverkehr bereit ist, lautet der Status InService. Die Spalte Health Status (Zustandsstatus) enthält das Ergebnis der Amazon EC2 Auto Scaling-Zustandsprüfungen für Ihre Instances. Obwohl eine Instance als fehlerfrei gekennzeichnet ist, sendet der Load Balancer Datenverkehr nur an Instances, welche die Load Balancer-Zustandsprüfungen bestehen.
5. Vergewissern Sie sich, dass Ihre Instances beim Load Balancer registriert sind. Öffnen Sie die [Zielgruppen-Seite](#) der Amazon EC2-Konsole. Wählen Sie Ihre Zielgruppe und danach die Registerkarte Ziele aus. Wenn der Zustand der Instances `initial` ist, liegt das wahrscheinlich daran, dass sie noch im Prozess der Registrierung sind, oder sie werden immer noch einer Zustandsprüfung unterzogen. Wenn der Zustand der Instances `healthy` ist, sind sie bereit zur Verwendung.

Schritt 4: Nächste Schritte

Nachdem Sie nun dieses Tutorial abgeschlossen haben, können Sie hier mehr erfahren:

- Amazon EC2 Auto Scaling bestimmt anhand des Status der von Ihrer Auto-Scaling-Gruppe verwendeten Zustandsprüfungen, ob eine Instance fehlerfrei ist. Wenn Sie Load Balancer-Integritätsprüfungen aktivieren und eine Instance die Integritätsprüfungen nicht besteht, betrachtet Ihre Auto Scaling Scaling-Gruppe die Instance als fehlerhaft und ersetzt sie. Weitere Informationen finden Sie unter [Health checks \(Zustandsprüfungen\)](#).
- Sie können Ihre Anwendung auf eine zusätzliche Availability Zone in derselben Region erweitern, um die Fehlertoleranz im Falle einer Service-Unterbrechung zu erhöhen. Weitere Informationen finden Sie unter [Fügen Sie eine Availability Zone hinzu](#).
- Sie können die Auto-Scaling-Gruppe für die Verwendung einer Ziel-Nachverfolgung-Skalierungsrichtlinie konfigurieren. Dadurch wird die Anzahl der Instances automatisch erhöht oder verringert, wenn sich der Bedarf für Ihre Instances ändert. Auf diese Weise kann die Gruppe Änderungen an der Menge des Datenverkehrs verarbeiten, den Ihre Anwendung erhält. Weitere Informationen finden Sie unter [Skalierungsrichtlinien für die Ziel-Nachverfolgung](#).

Schritt 5: Bereinigen

Nachdem Sie mit den für diese praktische Anleitung erstellten Ressourcen abgeschlossen haben, sollten Sie sie bereinigen, um unnötige Kosten zu vermeiden.

So löschen Sie Ihre Auto-Scaling-Gruppe

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.
3. Wählen Sie Löschen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **delete** ein, um das Löschen der angegebenen Auto-Scaling-Gruppe zu löschen, wählen Sie dann Löschen.

Ein Ladesymbol in der Spalte Name zeigt an, dass die Auto-Scaling-Gruppe gelöscht wird. Wenn der Löschvorgang erfolgt ist, zeigen die Spalten Desired (Gewünscht), Min und Max 0-Instances für die Auto-Scaling-Gruppe an. Es dauert einige Minuten, bis die Instance beendet und die Gruppe gelöscht werden. Aktualisieren Sie die Liste, um den aktuellen Status anzuzeigen.

Überspringen Sie folgenden Schritte, falls Sie die Startvorlage behalten möchten.

So löschen Sie eine Startvorlage

1. Öffnen Sie die Seite [Startvorlagen](#) der Amazon EC2 Konsole.
2. Wählen Sie Ihre Startvorlage aus.
3. Wählen Sie Actions (Aktionen) und Delete template (Vorlage löschen) aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **Delete** ein, um das Löschen der angegebenen Auto-Scaling-Gruppe zu bestätigen, wählen Sie dann Löschen.

Überspringen Sie folgenden Schritte, falls Sie die Startkonfiguration behalten möchten.

Löschen Sie eine Startkonfiguration wie folgt:

1. Öffnen Sie die Seite [Konfigurationen starten](#) der Amazon EC2-Konsole.
2. Wählen Sie Ihre Startkonfiguration aus.
3. Wählen Sie Actions, Delete launch configuration aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Delete (Löschen).

Überspringen Sie das folgende Verfahren, wenn Sie den Load Balancer für eine spätere Verwendung behalten möchten.

Löschen Sie den Load Balancer wie folgt:

1. Öffnen Sie die Seite [Load Balancers](#) in der Amazon EC2-Konsole.
2. Wählen Sie den Load Balancer aus und klicken Sie auf Actions (Aktionen) und dann auf Delete (Löschen).
3. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Yes, Delete (Ja, löschen).

So löschen Sie Ihre Zielgruppe

1. Öffnen Sie die [Zielgruppen-Seite](#) der Amazon EC2-Konsole.
2. Wählen Sie Ihre Zielgruppe aus und klicken Sie dann auf Actions (Aktionen), Delete (Löschen).
3. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Ja, löschen.

Zugehörige Ressourcen

Mit CloudFormation können Sie AWS Infrastrukturbereitstellungen vorhersehbar und wiederholt erstellen und bereitstellen, indem Sie mithilfe von Vorlagendateien eine Sammlung von Ressourcen als eine Einheit (einen Stapel) erstellen und löschen. Weitere Informationen finden Sie im [AWS CloudFormation -Benutzerhandbuch](#).

Eine exemplarische Vorgehensweise, die Ihnen zeigt, wie Sie mit einer Stack-Vorlage eine Auto-Scaling-Gruppe hinter einem Application Load Balancer bereitstellen, finden Sie unter [Exemplarische Vorgehensweise: Erstellen einer skalierten Anwendung mit Lastenausgleich](#) im AWS CloudFormation -Benutzerhandbuch. Verwenden Sie die exemplarische Vorgehensweise und die Mustervorlagen als Ausgangspunkt für die Erstellung ähnlicher Vorlagen, die Ihren Anforderungen entsprechen.

Auto Scaling Scaling-Startvorlagen

Eine Startvorlage ist ähnlich wie eine [Startkonfiguration](#) und gibt wie diese Instance-Konfigurationsinformationen an. Enthalten sind die ID des Amazon Machine Image (AMI), der Instance-Typ, ein Schlüsselpaar, Sicherheitsgruppen und die anderen Parameter, die Sie zum Starten von EC2-Instances verwenden. Durch das Definieren einer Startvorlage anstelle einer Startkonfiguration sind jedenfalls mehrere Versionen einer Startvorlage möglich.

Mit dem Versioning von Startvorlagen können Sie eine Teilmenge des vollständigen Satzes an Parametern erstellen. Anschließend können Sie es erneut verwenden, um andere Versionen derselben Startvorlage zu erstellen. Sie können beispielsweise eine Startvorlage erstellen, die eine Basiskonfiguration ohne AMI- oder Benutzerdatenskript definiert. Nachdem Sie Ihre Startvorlage erstellt haben, können Sie eine neue Version erstellen und die AMI- und Benutzerdaten mit der neuesten Version Ihrer Anwendung zum Testen hinzufügen. Dies führt zu zwei Versionen der Startvorlage. Das Speichern einer Basiskonfiguration hilft Ihnen, die erforderlichen allgemeinen Konfigurationsparameter beizubehalten. Sie können eine neue Version Ihrer Startvorlage aus der Basiskonfiguration erstellen, wann immer Sie möchten. Sie können auch die Versionen löschen, die zum Testen Ihrer Anwendung verwendet werden, wenn Sie sie nicht mehr benötigen.

Es wird empfohlen, Startvorlagen zu verwenden, um sicherzustellen, dass Sie auf die neuesten Funktionen und Verbesserungen zugreifen. Nicht alle Amazon EC2 Auto Scaling-Funktionen sind verfügbar, wenn Sie Startkonfigurationen verwenden. Sie können beispielsweise keine Auto-Scaling-Gruppe erstellen, die Spot- und On-Demand-Instances startet oder mehrere Instance-Typen angibt. Sie müssen eine Startvorlage verwenden, um diese Funktionen zu konfigurieren. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

Mit Startvorlagen können Sie auch neuere Funktionen von Amazon EC2 verwenden. Dazu gehören Systems Manager Manager-Parameter (AMI-ID), die aktuelle Generation der bereitgestellten EBS IOPS-Volumes (io2), EBS-Volume-Tagging, T2 Unlimited-Instances, Kapazitätsreservierungen und Dedicated HostsCapacity Blocks, um nur einige zu nennen.

Beim Erstellen einer Startvorlage sind alle Parameter optional. Wenn eine Startvorlage jedoch kein AMI angibt, können Sie das AMI nicht hinzufügen, wenn Sie Ihre Auto-Scaling-Gruppe erstellen. Wenn Sie ein AMI angeben, aber keinen Instance-Typ haben, können Sie beim Erstellen der Auto-Scaling-Gruppe einen oder mehrere Instance-Typen hinzufügen.

Inhalt

- [Berechtigungen für die Arbeit mit Startvorlagen](#)
- [Von Startvorlagen unterstützte API-Operationen](#)
- [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#)
- [Erstellen einer Startvorlage mithilfe erweiterter Einstellungen](#)
- [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#)
- [Migrieren Sie AWS CloudFormation Stacks zu Startvorlagen](#)
- [Beispiele für die Erstellung und Verwaltung von Startvorlagen mit dem AWS CLI](#)
- [Verwenden Sie AWS Systems Manager Parameter anstelle von AMI IDs in Startvorlagen](#)

Berechtigungen für die Arbeit mit Startvorlagen

Bei den Verfahren in diesem Abschnitt wird davon ausgegangen, dass Sie bereits über die erforderlichen Berechtigungen zum Erstellen von Startvorlagen verfügen. Informationen darüber, wie ein Administrator Ihnen Berechtigungen erteilt, finden Sie unter [Steuern des Zugriffs auf Startvorlagen mit IAM-Berechtigungen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Wenn Sie nicht über ausreichende Berechtigungen für die Verwendung und Erstellung von Ressourcen verfügen, die in einer Startvorlage angegeben sind, erhalten Sie eine Fehlermeldung, dass Sie nicht berechtigt sind, die Startvorlage zu verwenden, wenn Sie versuchen, sie für eine Auto-Scaling-Gruppe zu spezifizieren. Weitere Informationen finden Sie unter [Fehlersuche bei Amazon EC2 Auto Scaling: Startvorlagen](#).

Beispiele für IAM-Richtlinien, mit denen Sie die RunInstances API-Operationen `CreateAutoScalingGroup` `UpdateAutoScalingGroup`, und mit einer Startvorlage aufrufen können, finden Sie unter [Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling-Gruppen](#)

Von Startvorlagen unterstützte API-Operationen

Eine Liste der API-Operationen, die von Startvorlagen unterstützt werden, finden Sie unter [Amazon-EC2-Aktionen](#) in der [Amazon-EC2-API-Referenz](#).

Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe

Bevor Sie eine Auto-Scaling-Gruppe mit einer Startvorlage erstellen können, müssen Sie eine Startvorlage erstellen, die die Konfigurationsinformationen zum Starten einer Instance enthält, einschließlich der ID des Amazon Machine Image (AMI).

Verwenden Sie die folgenden Verfahren, um neue Startvorlagen zu erstellen.

Inhalt

- [So erstellen Sie eine Startvorlage \(Konsole\)](#)
- [So ändern Sie die Standardeinstellungen für die Netzwerkschnittstelle \(Konsole\)](#)
- [Ändern Sie die Speicherkonfiguration \(Konsole\)](#)
- [Erstellen Sie eine Startvorlage anhand einer vorhandenen Instance \(Konsole\)](#)
- [Zugehörige Ressourcen](#)
- [Einschränkungen](#)

Important

Die Parameter der Startvorlage werden nicht vollständig validiert, wenn Sie die Startvorlage erstellen. Wenn Sie falsche Werte für Parameter angeben oder keine unterstützten Parameterkombinationen verwenden, können keine Instances unter Verwendung dieser Startvorlage gestartet werden. Stellen Sie sicher, dass Sie die richtigen Werte für die Parameter angeben, und verwenden Sie unterstützte Parameterkombinationen. Um beispielsweise Instances mit einem Arm-basierten AWS Graviton- oder Graviton2-AMI zu starten, müssen Sie einen Arm-kompatiblen Instance-Typ angeben. Weitere Informationen finden Sie unter [Einschränkungen für Startvorlagen](#) im Amazon EC2 EC2-Benutzerhandbuch.

So erstellen Sie eine Startvorlage (Konsole)

In den folgenden Schritten wird beschrieben, wie Sie eine einfache Startvorlage konfigurieren:

- Geben Sie das Amazon Machine Image (AMI) an, von dem aus die Instances gestartet werden sollen.
- Wählen Sie einen Instance-Typ aus, der mit dem von Ihnen angegebenen AMI kompatibel ist.

- Geben Sie das Schlüsselpaar an, das Sie bei der Verbindung mit Instances verwenden möchten, z. B. mithilfe von SSH.
- Fügen Sie eine oder mehrere Sicherheitsgruppen hinzu, um Netzwerkzugriff auf die Instances zuzulassen.
- Geben Sie an, ob jeder Instance zusätzliche Volumes zugeordnet werden sollen.
- Fügen Sie benutzerdefinierte Tags (Schlüssel-Wert-Paare) zu den Instances und Volumes hinzu.

Eine Startvorlage erstellen

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich unter Instances die Option Launch Templates aus.
3. Wählen Sie Startvorlage erstellen. Geben Sie einen Namen und eine Beschreibung für die anfängliche Version der Startvorlage ein.
4. (Optional) Aktivieren Sie das Kontrollkästchen unter Auto Scaling-Anleitung, damit Amazon EC2 eine hilfreiche Anleitung bei der Erstellung einer Vorlage für die Verwendung mit Amazon EC2 Auto Scaling bereitstellt.
5. Füllen Sie unter Launch template contents (Vorlageninhalte starten), jedes erforderliche Feld und alle optionalen Felder aus.
 - a. Application and OS Images (Amazon Machine Image) (Anwendungs- und Betriebssystem-Images (Amazon Machine Image)): (Erforderlich) Wählen Sie die ID des AMI für Ihre Instances aus. Sie können alle verfügbaren AMIs durchsuchen oder ein AMI aus der Liste „Zuletzt verwendet“ oder „Schnellstart“ auswählen. Wenn Sie das benötigte AMI nicht sehen, wählen Sie Browse more, AMIs um den vollständigen AMI-Katalog zu durchsuchen.

Um ein benutzerdefiniertes AMI auszuwählen, müssen Sie zuerst Ihr AMI aus einer benutzerdefinierten Instance erstellen. Weitere Informationen finden Sie unter [Erstellen eines Amazon EBS-backed AMI](#) im Amazon EC2 EC2-Benutzerhandbuch.

- b. Wählen Sie für Instance type (Instance-Type) einen einzelnen Instance-Typ aus, der mit dem von Ihnen angegebenen AMI kompatibel ist.

Wenn Sie alternativ die attributbasierte Auswahl des Instance-Typs verwenden möchten, wählen Sie Erweitert, Spezifizieren von Instance-Typen-Attributen aus und geben Sie die folgenden Optionen an:

- Anzahl von v CPUs: Geben Sie die Mindest- und Höchstzahl von v ein. CPUs Um keine Grenzwerte anzugeben, geben Sie einen Mindestwert von 0 ein und lassen Sie den Höchstwert leer.
 - Amount of memory (MiB) (Speichermenge (MiB)): Geben Sie in MiB den minimalen und maximalen Speicher für Ihre Rechenanforderungen ein. Um keine Limits anzugeben, geben Sie mindestens 0 ein und lassen Sie das Maximum leer.
 - Erweitern Sie Optional instance type attributes (Optionale Instance-Typ-Attribute) und wählen Sie Add attribute (Attribut hinzufügen), um die Arten von Instances, die zur Erreichung Ihrer gewünschten Kapazität verwendet werden können, weiter einzuschränken. Informationen zu den einzelnen Attributen finden Sie [InstanceRequirementsRequest](#) in der Amazon EC2 API-Referenz.
 - Resultierende Instance-Typen: Sie können die Instance-Typen anzeigen, die den angegebenen Rechenanforderungen entsprechen, z. B. vCPUs, Arbeitsspeicher und Speicher.
 - Um Instance-Typen auszuschließen, wählen Sie Add Attribut (Attribut hinzufügen). Wählen Sie in der Liste Attribute (Attribut) Excluded instance types (Ausgeschlossene Instance-Typen). Wählen Sie aus der Liste Attributwert die auszuschließenden Instance-Typen aus.
- c. Key pair (login) (Schlüsselpaar (Login)): Wählen Sie für Key pair name (Schlüsselpaarname) ein vorhandenes Schlüsselpaar aus oder wählen Sie Create new key pair (Neues Schlüsselpaar erstellen), um ein neues zu erstellen. Weitere Informationen finden Sie unter [Amazon EC2 EC2-Schlüsselpaare und Linux-Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
- d. Network settings (Netzwerkeinstellungen): Wählen Sie für Firewall (security groups) (Firewall (Sicherheitsgruppen)) eine oder mehrere Sicherheitsgruppen aus, oder lassen Sie sie leer und konfigurieren Sie eine oder mehrere Sicherheitsgruppen als Teil der Netzwerkschnittstelle. Weitere Informationen finden Sie unter [Amazon-EC2-Sicherheitsgruppen für Linux-Instances](#) im Amazon-EC2-Benutzerhandbuch.

Wenn Sie in Ihrer Startvorlage keine Sicherheitsgruppen angeben, verwendet Amazon EC2 die Standardsicherheitsgruppe für die VPC, in der Ihre Auto-Scaling-Gruppe Instances startet. Standardmäßig lässt diese Sicherheitsgruppe eingehenden Datenverkehr von externen Netzwerken nicht zu. Weitere Informationen finden Sie unter [Standardsicherheitsgruppen für Sie VPCs](#) im Amazon VPC-Benutzerhandbuch.

- e. Führen Sie eine der folgenden Aktionen aus:

- So ändern Sie die Standardeinstellungen für die Netzwerkschnittstelle. Sie können beispielsweise die Funktion IPv4 zur öffentlichen Adressierung aktivieren oder deaktivieren, wodurch die Einstellung für die automatische Zuweisung öffentlicher IPv4 Adressen im Subnetz außer Kraft gesetzt wird. Weitere Informationen finden Sie unter [So ändern Sie die Standardeinstellungen für die Netzwerkschnittstelle \(Konsole\)](#).
 - Überspringen Sie diesen Schritt, um die Standardeinstellungen für die Netzwerkschnittstelle beizubehalten.
- f. Führen Sie eine der folgenden Aktionen aus:
- Ändern Sie die Speicherkonfiguration. Weitere Informationen finden Sie unter [Ändern Sie die Speicherkonfiguration \(Konsole\)](#).
 - Überspringen Sie diesen Schritt, um die Standardspeicherkonfiguration beizubehalten.
- g. Geben Sie für Resource Tags (Ressourcen-Tags) die Tags an, indem Sie Schlüssel-Wert-Kombinationen bereitstellen. Wenn Sie Instance-Tags in Ihrer Startvorlage angeben und sich dann dafür entschieden haben, die Tags Ihrer Auto-Scaling-Gruppe an ihre Instances zu übertragen, werden alle Tags zusammengeführt. Wenn derselbe Tag-Schlüssel für einen Tag in Ihrer Startvorlage und einen Tag in Ihrer Auto-Scaling-Gruppe angegeben wird, hat der Tag-Wert aus der Gruppe Vorrang.
6. (Optional) Konfigurieren erweiterter Einstellungen. Beispielsweise können Sie eine IAM-Rolle auswählen, die Ihre Anwendung verwenden kann, wenn sie auf andere AWS -Ressourcen zugreift. Alternativ können Sie die Instance-Benutzerdaten angeben, die zum Ausführen allgemeiner automatisierter Konfigurationsaufgaben nach dem Start einer Instance verwendet werden können. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage mithilfe erweiterter Einstellungen](#).
7. Wenn Sie bereit sind, die Startvorlage zu erstellen, wählen Sie Create launch template (Startvorlage erstellen) aus.
8. Wählen Sie auf der Bestätigungsseite Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus, um eine Auto-Scaling-Gruppe zu erstellen.

So ändern Sie die Standardeinstellungen für die Netzwerkschnittstelle (Konsole)

Netzwerkschnittstellen bieten eine Konnektivität zu anderen Ressourcen in Ihrer VPC und im Internet. Weitere Informationen finden Sie unter [Stellen Sie Netzwerkkonnektivität für Ihre Auto-Scaling-Instances mit Amazon VPC bereit](#).

In diesem Abschnitt erfahren Sie, wie Sie die Standardeinstellungen für die Netzwerkschnittstelle ändern. Sie können beispielsweise definieren, ob Sie jeder Instance eine öffentliche IPv4 Adresse zuweisen möchten, anstatt standardmäßig die Einstellung für die automatische Zuweisung öffentlicher IPv4 Adressen im Subnetz zu verwenden.

Überlegungen und Einschränkungen

Beachten Sie beim Ändern der Standardeinstellungen für die Netzwerkschnittstelle die folgenden Überlegungen und Einschränkungen:

- Sie müssen die Sicherheitsgruppen als Teil der Netzwerkschnittstelle angeben und nicht im Bereich Security Groups (Sicherheitsgruppen) der Vorlage. Sie können keine Sicherheitsgruppen an beiden Orten festlegen.
- Wenn Sie eine vorhandene Netzwerkschnittstellen-ID angeben, können Sie nur eine Instance starten. Dazu müssen Sie das AWS CLI oder ein SDK verwenden, um die Auto Scaling Scaling-Gruppe zu erstellen. Wenn Sie die Gruppe erstellen, müssen Sie die Availability Zone angeben, jedoch nicht die Subnetz-ID. Außerdem können Sie eine vorhandene Netzwerkschnittstelle nur angeben, wenn sie einen Geräteindex von 0 hat.
- Sie können eine öffentliche IPv4 Adresse nicht automatisch zuweisen, wenn Sie mehr als eine Netzwerkschnittstelle angeben. Sie können auch keine doppelten Geräteindizes über Netzwerkschnittstellen hinweg angeben. Sowohl die primäre als auch die sekundäre Netzwerkschnittstelle befinden sich im selben Subnetz.
- Wenn eine Instance gestartet wird, wird jeder Netzwerkschnittstelle automatisch eine private Adresse zugewiesen. Die Adresse stammt aus dem CIDR-Bereich des Subnetzes, in dem die Instance gestartet wird. Weitere Informationen zum Angeben von CIDR-Blöcken (oder IP-Adressbereichen) für Ihre VPC oder ein Subnetz finden Sie im [Amazon-VPC-Benutzerhandbuch](#).

So ändern Sie die Standardeinstellungen für die Netzwerkschnittstelle

1. Erweitern Sie unter Network settings (Netzwerkeinstellungen) Advanced network configuration (Erweiterte Netzwerkkonfiguration).
2. Wählen Sie Add network interface (Netzwerkschnittstelle hinzufügen) aus, um die primäre Netzwerkschnittstelle zu konfigurieren, und beachten Sie dabei die folgenden Felder:
 - a. Device index (Geräteindex): Behalten Sie den Standardwert 0 bei, um Ihre Änderungen auf die primäre Netzwerkschnittstelle (eth0) anzuwenden.
 - b. Network interface (Netzwerkschnittstelle): Behalten Sie den Standardwert New interface (Neue Schnittstelle) bei, um Amazon EC2 Auto Scaling automatisch eine neue Netzwerkschnittstelle erstellen zu lassen, wenn eine Instance gestartet wird. Alternativ können Sie eine vorhandene, verfügbare Netzwerkschnittstelle mit einem Geräteindex von 0 auswählen, dies beschränkt Ihre Auto-Scaling-Gruppe jedoch auf eine Instance.
 - c. Description (Beschreibung): (Optional) Geben Sie einen beschreibenden Namen ein.
 - d. Subnet (Subnetz): Behalten Sie die Standardeinstellung Don't include in launch template (Nicht in die Startvorlage einschließen) bei.

Wenn das AMI ein Subnetz für die Netzwerkschnittstelle angibt, führt dies zu einem Fehler. Es wird empfohlen, Auto Scaling guidance (Anleitung zur automatischen Skalierung) als Problemumgehung zu deaktivieren. Nachdem Sie diese Änderung vorgenommen haben, erhalten Sie keine Fehlermeldung. Unabhängig davon, wo das Subnetz angegeben ist, haben die Subnetzeinstellungen der Auto-Scaling-Gruppe Vorrang und können nicht überschrieben werden.

- e. Öffentliche IP automatisch zuweisen: Ändern Sie, ob Ihre Netzwerkschnittstelle mit einem Geräteindex von 0 eine öffentliche IPv4 Adresse erhält. Standardmäßig erhalten Instances in einem Standard-Subnetz eine öffentliche IPv4 Adresse, Instances in einem nicht standardmäßigen Subnetz dagegen nicht. Wählen Sie Enable oder Disable aus, um die Standardeinstellungen des Subnetzes zu überschreiben.
- f. Security groups (Sicherheitsgruppen): Wählen Sie eine oder mehrere Sicherheitsgruppen für die Netzwerkschnittstelle aus. Jede Sicherheitsgruppe muss für die VPC konfiguriert werden, in der Ihre Auto-Scaling-Gruppe Instances starten wird. Weitere Informationen finden Sie unter [Amazon-EC2-Sicherheitsgruppen für Linux-Instances](#) im Amazon-EC2-Benutzerhandbuch.

- g. Delete on termination (Beim Beenden löschen): Wählen Sie Yes (Ja), um die Netzwerkschnittstelle zu löschen, wenn die Instance beendet wird, oder wählen Sie No (Nein), um die Netzwerkschnittstelle beizubehalten.
 - h. Elastic Fabric Adapter: Um Anwendungsfälle für High Performance Computing und Machine Learning zu unterstützen, ändern Sie die Netzwerkschnittstelle in eine Elastic-Fabric-Adapter-Netzwerkschnittstelle. Weitere Informationen finden Sie unter [Elastic Fabric Adapter](#) im Amazon EC2 EC2-Benutzerhandbuch.
 - i. Network card index (Netzwerkkarten-Index): Wählen Sie 0 aus, um die primäre Netzwerkschnittstelle mit einem Geräteindex von 0 an die Netzwerkkarte anzuschließen. Wenn diese Option nicht verfügbar ist, behalten Sie den Standardwert Don't include in launch template (Nicht in die Startvorlage einschließen) bei. Das Anschließen der Netzwerkschnittstelle an eine bestimmte Netzwerkkarte ist nur für unterstützte Instance-Typen verfügbar. Weitere Informationen finden Sie unter [Netzwerkkarten](#) im Amazon EC2 EC2-Benutzerhandbuch.
 - j. ENA Express: Wählen Sie für Instance-Typen, die ENA Express unterstützen, Enable, um ENA Express zu aktivieren, oder Disable, um ENA Express zu deaktivieren. Weitere Informationen finden Sie unter [Verbessern der Netzwerkleistung mit ENA Express auf Linux-Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
 - k. ENA Express UDP: Wenn Sie ENA Express aktivieren, können Sie es optional für UDP-Traffic verwenden. Wählen Sie Aktivieren, um ENA Express UDP zu aktivieren, oder Deaktivieren, um es zu deaktivieren.
3. Wählen Sie zum Hinzufügen einer sekundären Netzwerkschnittstelle Netzwerkschnittstelle hinzufügen aus.

Ändern Sie die Speicherkonfiguration (Konsole)

Sie können die Speicherkonfiguration für Instances ändern, die von einem Amazon-EBS-gestützten AMI oder einem Instance-Speicher-gestützten AMI gestartet werden. Sie können auch zusätzliche EBS-Volumes angeben, die an die Instances angefügt werden sollen. Das AMI enthält ein oder mehrere Speicher-Volumes, einschließlich des Root-Volumes (Volume 1 (AMI Root)).

So ändern Sie die Speicherkonfiguration

1. In `:Configure storage` (Speicher konfigurieren), ändern Sie die Größe oder den Typ des Volumes.

Wenn der Wert, den Sie für die Volumegröße angeben, außerhalb der Grenzwerte des Volume-Typs liegt oder kleiner als die Snapshotgröße ist, wird eine Fehlermeldung angezeigt. Um Ihnen bei der Behebung des Problems zu helfen, gibt diese Nachricht den minimalen oder maximalen Wert an, den das Feld akzeptieren kann.

Es werden nur Volumes angezeigt, die einem Amazon-EBS-gestützten AMI zugeordnet sind. Um Informationen zur Speicherkonfiguration für eine Instance anzuzeigen, die von einem Instance-Speicher-gestützten AMI gestartet wurde, wählen Sie Show details (Details anzeigen) aus dem Abschnitt Instance store volumes (Instance-Speicher-Volumes) aus.

Um alle EBS-Volume-Parameter anzugeben, wechseln Sie zur Advanced(Erweitert)-Ansicht in der rechten oberen Ecke.

2. Erweitern Sie für erweiterte Optionen das Volume, das Sie ändern möchten, und konfigurieren Sie das Volume wie folgt:
 - a. Storage type (Speichertyp): Der Typ des Volumes (EBS oder flüchtig), das Ihrer Instance zugeordnet werden soll. Der Volume-Typ Instance-Speicher (flüchtig) ist nur verfügbar, wenn Sie einen Instance-Typ auswählen, der ihn unterstützt. Weitere Informationen finden Sie unter [Amazon EBS-Volumes](#) im Amazon EBS-Benutzerhandbuch und Amazon [EC2 EC2-Instance-Speicher im Amazon EC2](#) EC2-Benutzerhandbuch.
 - b. Device name (Gerätename): Wählen Sie aus der Liste verfügbarer Gerätenamen für das Volume einen Eintrag aus.
 - c. Snapshot: Wählen Sie den Snapshot aus, von dem das Volume erstellt werden soll. Sie können nach verfügbaren freigegebenen und öffentlichen Snapshots suchen, indem Sie Text in das Feld Snapshot eingeben.
 - d. Größe (GiB): Sie können für EBS-Volumes eine Speichergröße angeben. Wenn Sie ein AMI und eine Instance ausgewählt haben, die im kostenlosen Kontingent enthalten sind, dürfen Sie den Grenzwert von 30 GiB Gesamtspeicher nicht überschreiten, um innerhalb des kostenlosen Kontingents zu bleiben. Weitere Informationen finden Sie unter [Einschränkungen für die Größe und Konfiguration eines EBS-Volumes](#) im Amazon EBS-Benutzerhandbuch.
 - e. Volume type (Volume-Typ): Wählen Sie für EBS-Volumes den Volume-Typ aus. Weitere Informationen finden Sie unter [Amazon-EBS-Volume-Typen](#) im Amazon-EBS-Benutzerhandbuch.
 - f. IOPS: Wenn Sie den Volumetyp Provisioned IOPS SSD (io1undio2) oder General Purpose SSD (gp3) ausgewählt haben, können Sie die Anzahl der I/O Operationen pro

Sekunde (IOPS) eingeben, die das Volume unterstützen kann. Dies ist für io1-, io2- und gp3-Volumes erforderlich. Wird bei gp2-, st1-, sc1- oder Standard-Volumes nicht unterstützt.

- g. Delete on termination (Bei Beendigung löschen): Wählen Sie für EBS-Volumes Yes (Ja), um das Volume zu löschen, wenn die Instance beendet wird, oder wählen Sie No (Nein), um das Volume beizubehalten.
- h. Encrypted (Verschlüsselt): Wenn der Instance-Typ die EBS-Verschlüsselung unterstützt, können Sie Ja auswählen, um die Verschlüsselung für das Volume zu aktivieren. Wenn Sie für diese Region die standardmäßige Verschlüsselung aktiviert haben, wird der Standard-CMK für Sie ausgewählt. Weitere Informationen finden Sie unter [Amazon EBS-Verschlüsselung](#) und [Amazon EBS-Verschlüsselung standardmäßig aktivieren](#) im Amazon EBS-Benutzerhandbuch.

Der Standardeffekt bei diesem Parameter hängt von der Wahl der Volume-Quelle ab, wie in der folgenden Tabelle beschrieben. In jedem Fall müssen Sie über die Erlaubnis verfügen, die angegebenen Daten zu verwenden. AWS KMS key

Verschlüsselungsergebnisse

Wenn für den Parameter Encrypted Folgendes festgelegt ist ...	Und wenn die Quelle des Volumes Folgendes ist ...	Dann ist der Standardverschlüsselungsstatus ...	Hinweise
Nein	Neues (leeres) Volume	Unverschlüsselt*	–
	Unverschlüsselter eigener Snapshot	Unverschlüsselt*	
	Verschlüsselter eigener Snapshot	Verschlüsselt mit demselben Schlüssel	
	Unverschlüsselter Snapshot, der mit Ihnen geteilt wird	Unverschlüsselt*	

Wenn für den Parameter Encrypted Folgendes festgelegt ist ...	Und wenn die Quelle des Volumes Folgendes ist ...	Dann ist der Standardverschlüsselungsstatus ...	Hinweise
	Verschlüsselter Snapshot, der mit Ihnen geteilt wird	Verschlüsselt durch standardmäßigen KMS-Schlüssel	
Ja	Neues Volume	Verschlüsselt durch standardmäßigen KMS-Schlüssel	Um einen nicht standardmäßigen KMS-Schlüssel zu verwenden, geben Sie einen Wert für den KMS key(KMS-Schlüssel)-Parameter ein.
	Unverschlüsselter eigener Snapshot	Verschlüsselt durch standardmäßigen KMS-Schlüssel	
	Verschlüsselter eigener Snapshot	Verschlüsselt mit demselben Schlüssel	
	Unverschlüsselter Snapshot, der mit Ihnen geteilt wird	Verschlüsselt durch standardmäßigen KMS-Schlüssel	
	Verschlüsselter Snapshot, der mit Ihnen geteilt wird	Verschlüsselt durch standardmäßigen KMS-Schlüssel	

* Wenn die standardmäßige Verschlüsselung aktiviert ist, werden alle neu erstellten Volumes (unabhängig davon, ob der Encrypted(Verschlüsselt)-Parameter auf Yes (Ja) gesetzt ist) mit dem standardmäßigen KMS-Schlüssel verschlüsselt. Wenn Sie sowohl den Encrypted(Verschlüsselt)- als auch den Key(Schlüssel)-Parameter festlegen, können Sie einen nicht standardmäßigen KMS-Schlüssel angeben.

- i. KMS key (KMS-Schlüssel): Wenn Sie Yes (Ja) für Encrypted (Verschlüsselt) auswählen, müssen Sie einen kundenverwalteten Schlüssel zum Verschlüsseln des Volumes

auswählen. Wenn die für diese Region die standardmäßige Verschlüsselung aktiviert haben, wird der Standard-CMK für Sie ausgewählt. Sie können einen anderen Schlüssel auswählen oder den ARN eines beliebigen kundenverwalteten Schlüssels angeben, den Sie zuvor mit dem AWS Key Management Service erstellt haben.

3. Um zusätzliche Volumes anzugeben, die an die von dieser Startvorlage gestarteten Instances angehängt werden sollen, wählen Sie Add new volume (Neues Volume hinzufügen) aus.

Erstellen Sie eine Startvorlage anhand einer vorhandenen Instance (Konsole)

So erstellen Sie eine Startvorlage anhand einer vorhandenen Instance

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich unter Instances (Instances) die Option Instances (Instances) aus.
3. Wählen Sie die Instance und anschließend Aktionen, Image und Vorlagen, Eine Vorlage aus einer Instance erstellen aus.
4. Geben Sie einen Namen und eine Beschreibung ein.
5. Unter Auto-Scaling-Anleitung aktivieren Sie das Kontrollkästchen.
6. Passen Sie alle Einstellungen wie erforderlich an und wählen Sie Startvorlage erstellen aus.
7. Wählen Sie auf der Bestätigungsseite Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus, um eine Auto-Scaling-Gruppe zu erstellen.

Zugehörige Ressourcen

Wir stellen einige JSON- und YAML-Vorlagenausschnitte zur Verfügung, anhand derer Sie verstehen können, wie Sie Startvorlagen in Ihren CloudFormation Stack-Vorlagen deklarieren. Weitere Informationen finden Sie in den CloudFormation Abschnitten [AWS::EC2::LaunchTemplate](#) und [Erstellen von Startvorlagen mit Hilfe](#) des AWS CloudFormation Benutzerhandbuchs.

Weitere Informationen zu Startvorlagen finden Sie unter [Starten einer Instance aus einer Startvorlage](#) im Amazon EC2 EC2-Benutzerhandbuch.

Einschränkungen

- Sie können zwar ein Subnetz in einer Startvorlage spezifizieren, aber das ist nicht notwendig, wenn Sie die Startvorlage nur zum Erstellen von Auto-Scaling-Gruppen verwenden. Sie können das Subnetz für eine Auto-Scaling-Gruppe nicht durch Spezifizierung des Subnetzes in einer Startvorlage festlegen. Die Subnetze für die Auto-Scaling-Gruppe werden aus der eigenen Ressourcendefinition der Auto-Scaling-Gruppe übernommen.
- Weitere Einschränkungen für benutzerdefinierte Netzwerkschnittstellen finden Sie unter [So ändern Sie die Standardeinstellungen für die Netzwerkschnittstelle \(Konsole\)](#).

Erstellen einer Startvorlage mithilfe erweiterter Einstellungen

In diesem Thema wird beschrieben, wie Sie eine Startvorlage mit erweiterten Einstellungen aus dem AWS-Managementkonsole erstellen.

Um eine Startvorlage mit erweiterten Einstellungen zu erstellen

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich unter Instances die Option Launch Templates und dann Create Launch Template aus.
3. Konfigurieren Sie Ihre Startvorlage wie in den folgenden Themen beschrieben:
 - [Erforderliche Einstellungen](#)
 - [Erweiterte Einstellungen](#)
4. Wählen Sie Startvorlage erstellen.

Erforderliche Einstellungen

Wenn Sie eine Startvorlage erstellen, müssen Sie die folgenden erforderlichen Einstellungen angeben.

Name der Startvorlage

Geben Sie einen eindeutigen Namen ein, der die Startvorlage beschreibt.

Anwendungs- und Betriebssystem-Images (Amazon Machine Image)

Wählen Sie das Amazon Machine Image (AMI) aus, das Sie verwenden möchten. Sie können entweder nach dem AMI suchen oder nach dem AMI suchen, das Sie verwenden möchten. Wählen Sie für eine optimale Skalierungseffizienz ein benutzerdefiniertes AMI, das vollständig so konfiguriert ist, dass es eine Instance mit Ihrem Anwendungscode startet und beim Start nur wenige Änderungen erfordert.

Instance-Typ

Wählen Sie einen Instance-Typ, der mit Ihrem AMI kompatibel ist. Sie können das Hinzufügen eines Instance-Typs zu Ihrer Startvorlage überspringen, wenn Sie mehrere Instance-Typen verwenden möchten, die in die eigene Ressourcendefinition der Auto Scaling Scaling-Gruppe eingebettet sind. Ein Instanztyp ist nur erforderlich, wenn Sie nicht vorhaben, eine [gemischte Instanzgruppe](#) zu erstellen.

Erweiterte Einstellungen

Die erweiterten Einstellungen sind optional. Wenn Sie keine erweiterten Einstellungen konfigurieren, werden die spezifischen Funktionen nicht zu Ihren Instances hinzugefügt.

Erweitern Sie den Abschnitt Erweiterte Details, um die erweiterten Einstellungen anzuzeigen. In den folgenden Abschnitten werden die nützlichsten erweiterten Einstellungen beschrieben, auf die Sie sich bei der Erstellung einer Startvorlage für eine Auto Scaling Scaling-Gruppe konzentrieren sollten. Weitere Informationen finden Sie unter [Erweiterte Details](#) im Amazon EC2 EC2-Benutzerhandbuch.

IAM-Instance-Profil

Das Instanzprofil enthält die IAM-Rolle, die Sie verwenden möchten. Wenn Ihre Auto Scaling Scaling-Gruppe eine EC2-Instance startet, werden die in der zugehörigen IAM-Rolle definierten Berechtigungen Anwendungen gewährt, die auf der Instance ausgeführt werden. Weitere Informationen finden Sie unter [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#).

Termination protection

Wenn diese Funktion aktiviert ist, verhindert sie, dass Benutzer eine Instance mithilfe der Amazon EC2 EC2-Konsole, CLI-Befehlen und API-Operationen beenden. Der Kündigungsschutz bietet einen zusätzlichen Schutz vor einer versehentlichen Kündigung. Es verhindert nicht, dass Amazon EC2 Auto Scaling eine Instance beendet. Informationen zur Steuerung, welche Instances

Amazon EC2 Auto Scaling beenden kann, finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).

Detallierte Überwachung CloudWatch

Sie können eine detaillierte Überwachung für Ihre EC2-Instances aktivieren, damit sie in Intervallen von 1 Minute Metrikdaten CloudWatch an Amazon senden können. Standardmäßig senden EC2-Instances Metrikdaten in Intervallen von 5 Minuten CloudWatch an. Es fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter [Überwachung für Auto-Scaling-Instances konfigurieren](#).

Kreditspezifikation

Amazon EC2 bietet Burstable-Performance-Instances wie T2, T3 und T3a, die es Anwendungen ermöglichen, bei Bedarf die CPU-Basisleistung zu überschreiten. Standardmäßig können diese Instances für eine begrenzte Zeit ausgelastet werden, bevor ihre CPU-Auslastung gedrosselt wird. Sie können optional den Modus „Unlimited“ aktivieren, sodass die Instances so lange wie nötig über den Basiswert hinaus übersteigen können. Auf diese Weise können Anwendungen bei Bedarf eine hohe CPU-Leistung aufrechterhalten. Zusätzliche Gebühren können anfallen. Weitere Informationen finden Sie unter [Verwenden einer Auto Scaling Scaling-Gruppe zum Starten einer Burstable-Performance-Instance als Unlimited](#) im Amazon EC2 EC2-Benutzerhandbuch.

Name der Platzierungsgruppe

Sie können eine Platzierungsgruppe angeben und mithilfe einer Cluster- oder Partitionsstrategie beeinflussen, wie sich Ihre Instances physisch im AWS Rechenzentrum befinden. Für kleine Auto Scaling Scaling-Gruppen können Sie auch die Spread-Strategie verwenden. Weitere Informationen finden Sie unter [Placement-Gruppen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Bei der Verwendung von Platzierungsgruppen mit Auto Scaling Scaling-Gruppen sind einige Überlegungen zu beachten:

- Wenn eine Platzierungsgruppe sowohl in der Startvorlage als auch in der Auto Scaling Scaling-Gruppe angegeben ist, hat die Platzierungsgruppe für die Auto Scaling Scaling-Gruppe Vorrang.
- Seien Sie vorsichtig CloudFormation, wenn Sie in der Startvorlage eine Platzierungsgruppe definieren. Amazon EC2 Auto Scaling startet Instances in der angegebenen Platzierungsgruppe. Empfangen CloudFormation Sie jedoch keine Signale von diesen Instances, wenn Sie eine [UpdatePolicy](#) mit Ihrer Auto Scaling Scaling-Gruppe verwenden (obwohl sich dies in future ändern könnte).

Kaufoption

Sie können Spot-Instances anfordern wählen, um Spot-Instances zum Spot-Preis, begrenzt auf den On-Demand-Preis, anzufordern, und „Anpassen“ wählen, um die Standardeinstellungen für Spot-Instances zu ändern. Bei einer Auto-Scaling-Gruppe müssen Sie eine einmalige Anforderung ohne Enddatum angeben (Standardeinstellung). Weitere Informationen finden Sie unter [Spot-Instances für fehlertolerante und flexible Anwendungen anfordern](#). Diese Einstellung kann unter besonderen Umständen nützlich sein, aber im Allgemeinen ist es am besten, sie nicht festzulegen und stattdessen eine gemischte Instances-Gruppe zu erstellen. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

Wenn Sie in Ihrer Startvorlage eine Spot Instance-Anfrage angeben, können Sie keine gemischte Instances-Gruppe erstellen. Wenn Sie versuchen, eine Startvorlage zu verwenden, die Spot Instances mit einer gemischten Instance-Gruppe anfordert, erhalten Sie die folgende Fehlermeldung: `Incompatible launch template: You cannot use a launch template that is set to request Spot Instances (InstanceMarketOptions) when you configure an Auto Scaling group with a mixed instances policy. Add a different launch template to the group and try again.`

Capacity Reservation

Kapazitätsreservierungen ermöglichen es Ihnen, Kapazität für Ihre Amazon EC2 EC2-Instances in einer bestimmten Availability Zone für einen beliebigen Zeitraum zu reservieren. Weitere Informationen finden Sie unter [On-Demand Capacity Reservations](#) im Amazon-EC2-Benutzerhandbuch.

Sie können wählen, ob Sie Instances starten möchten in:

- jede offene Kapazitätsreservierung (Offen)
- eine bestimmte Kapazitätsreservierung (Ziel nach ID)
- eine Gruppe von Kapazitätsreservierungen (Ziel nach Gruppe)

Um auf eine bestimmte Kapazitätsreservierung abzielen zu können, muss der Instance-Typ in Ihrer Startvorlage mit dem Instance-Typ der Reservierung übereinstimmen. Wenn Sie Ihre Auto Scaling Scaling-Gruppe erstellen, verwenden Sie dieselbe Availability Zone wie die Kapazitätsreservierung. Je nachdem, AWS-Region was Sie wählen, können Sie sich stattdessen für einen Kapazitätsblock entscheiden. Weitere Informationen finden Sie unter [Capacity BlocksFür Machine-Learning-Workloads verwenden](#).

Informationen zur gezielten Ausrichtung auf eine Gruppe von Kapazitätsreservierungen finden Sie unter [Reservieren Sie Kapazität in bestimmten Availability Zones mit Kapazitätsreservierungen](#). Wenn Sie auf eine Gruppe von Kapazitätsreservierungen abzielen, können Sie die Kapazität auf mehrere Availability Zones verteilen, um die Ausfallsicherheit zu verbessern.

Tenancy

Amazon EC2 bietet drei Optionen für die Wartung Ihrer EC2-Instances:

- **Gemeinsam genutzt (gemeinsam genutzt)** — Mehrere AWS-Konten können sich dieselbe physische Hardware teilen. Dies ist die Standard-Tenancy-Option beim Starten einer Instance.
- **Dedizierte Instances (Dedicated)** — Ihre Instance wird auf Single-Tenant-Hardware ausgeführt. Kein anderer AWS Kunde teilt sich denselben physischen Server. Weitere Informationen finden Sie unter [Dedicated Instances](#) im Amazon EC2-Benutzerhandbuch.
- **Dedicated Hosts (Dedicated Host)** — Die Instance wird auf einem physischen Server ausgeführt, der nur für Sie reserviert ist. Die Verwendung von Dedicated Hosts macht es einfacher, Ihre eigenen Lizenzen (BYOL) mit speziellen Hardwareanforderungen für EC2 zu verwenden und Compliance-Anwendungsfälle zu erfüllen. Wenn Sie diese Option wählen, müssen Sie eine Host-Ressourcengruppe für die Tenancy-Host-Ressourcengruppe angeben. Weitere Informationen finden Sie unter [Dedicated Hosts](#) im Amazon EC2 EC2-Benutzerhandbuch.

Support für Dedicated Hosts ist nur verfügbar, wenn Sie eine Host-Ressourcengruppe angeben. Sie können keinen bestimmten Host als Ziel festlegen oder eine Host-Platzierungsaffinität verwenden.

- Wenn Sie versuchen, eine Startvorlage zu verwenden, die eine Host-ID angibt, erhalten Sie die folgende Fehlermeldung: `Incompatible launch template: Tenancy host ID is not supported for Auto Scaling`.
- Wenn Sie versuchen, eine Startvorlage zu verwenden, die die Hostplatzierungsaffinität angibt, wird die folgende Fehlermeldung angezeigt: `Incompatible launch template: Auto Scaling does not support host placement affinity`.

Hostressourcengruppe „Tenancy“

Mit AWS License Manager können Sie Ihre eigenen Lizenzen verwenden AWS und diese zentral verwalten. Eine Host-Ressourcengruppe ist eine Gruppe von Dedicated Hosts, die mit einer bestimmten License Manager Manager-Lizenzkonfiguration verknüpft sind. Host-Ressourcengruppen ermöglichen es Ihnen, EC2-Instances einfach auf Dedicated Hosts zu starten, die Ihren Anforderungen an die Softwarelizenzierung entsprechen. Sie müssen Dedicated

Hosts nicht vorab manuell zuweisen. Sie werden bei Bedarf automatisch erstellt. Beachten Sie, dass, wenn Sie ein AMI mit einer Lizenzkonfiguration verknüpfen, dieses AMI jeweils nur einer Host-Ressourcengruppe zugeordnet werden kann. Weitere Informationen finden Sie unter [Host-Ressourcengruppen AWS License Manager im License Manager Manager-Benutzerhandbuch](#).

Lizenzkonfigurationen

Mit dieser Einstellung können Sie eine Lizenzkonfiguration für Ihre Instances angeben, ohne deren Tenancy auf Dedicated Hosts zu beschränken. Die Lizenzkonfiguration verfolgt die auf den Instances bereitgestellten Softwarelizenzen, sodass Sie Ihre Lizenznutzung und die Einhaltung der Vorschriften überwachen können. Weitere Informationen finden Sie unter [Erstellen einer selbstverwalteten Lizenz im License Manager Manager-Benutzerhandbuch](#).

Metadaten verfügbar

Sie können wählen, ob Sie den Zugriff auf den HTTP-Endpunkt des Instanz-Metadatendienstes aktivieren oder deaktivieren möchten. Standardmäßig ist der HTTP-Endpunkt aktiviert. Wenn Sie den Endpunkt deaktivieren, wird der Zugriff auf Ihre Instance-Metadaten deaktiviert. Sie können angeben, dass die Bedingung IMDSv2 nur erforderlich sein soll, wenn der HTTP-Endpunkt aktiviert ist. Weitere Informationen finden Sie unter [Konfigurieren der Instance-Metadatenoptionen](#) im Amazon-EC2-Benutzerhandbuch.

Metadaten-Version

Sie können festlegen, dass bei der Anforderung von Instanz-Metadaten die Verwendung von Instanz-Metadatendienst Version 2 (IMDSv2) vorgeschrieben wird. Wenn Sie keinen Wert angeben, werden standardmäßig IMDSv1 sowohl als auch unterstützt IMDSv2. Weitere Informationen finden Sie unter [Konfigurieren der Instance-Metadatenoptionen](#) im Amazon-EC2-Benutzerhandbuch.

Hop-Limit für die Antwort auf Metadaten

Sie können die zulässige Anzahl von Netzwerk-Hops für das Metadaten-Token festlegen. Wenn Sie keinen Wert angeben, wird der Standard auf 1 festgelegt. Weitere Informationen finden Sie unter [Konfigurieren der Instance-Metadatenoptionen](#) im Amazon-EC2-Benutzerhandbuch.

Benutzerdaten

Sie können Ihre Instances anpassen und die Konfiguration beim Start abschließen, indem Sie Shell-Skripts oder Cloud-Init-Direktiven als Benutzerdaten angeben. Die Benutzerdaten werden beim ersten Start der Instanz ausgeführt, sodass Sie Anwendungen, Abhängigkeiten oder Anpassungen beim Start automatisch installieren können. Weitere Informationen finden Sie unter [Befehle auf Ihrer Linux-Instance beim Start ausführen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Wenn Sie umfangreiche Downloads oder komplexe Skripts haben, verlängert dies die Zeit, die benötigt wird, bis die Instance einsatzbereit ist. In diesem Fall müssen Sie möglicherweise einen Lifecycle-Hook konfigurieren, um zu verhindern, dass eine Instanz den InService Status erreicht, bis sie vollständig bereitgestellt ist. Weitere Informationen zum Hinzufügen eines Lifecycle-Hooks zu Ihrer Auto Scaling Scaling-Gruppe finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).

Spot-Instances für fehlertolerante und flexible Anwendungen anfordern

In Ihrer Startvorlage können Sie optional Spot-Instances ohne Enddatum oder Dauer anfordern. Amazon EC2-Spot-Instances sind freie Kapazitäten mit hohen Rabatten im Vergleich zum EC2 On-Demand-Preis verfügbar. Spot Instances sind eine kostengünstige Wahl, sofern Sie bei der Ausführung Ihrer Anwendungen zeitlich flexibel sind und Unterbrechungen verschmerzen können. Weitere Informationen zum Erstellen einer Startvorlage, die Spot-Instanzen anfordert, finden Sie unter [Erstellen einer Startvorlage mithilfe erweiterter Einstellungen](#).


Important

Normalerweise werden Spot-Instances zur Ergänzung von On-Demand-Instances verwendet. In diesem Szenario können Sie die gleichen Einstellungen, die auch für den Start von Spot-Instances verwendet werden, als Teil der Einstellungen Ihrer Auto-Scaling-Gruppe festlegen. Wenn Sie die Einstellungen als Teil der Auto-Scaling-Gruppe angeben, können Sie Spot-Instances erst nach dem Start einer bestimmten Anzahl von On-Demand-Instances starten und dann eine Kombination aus On-Demand-Instances und Spot-Instances starten, während die Gruppe skaliert wird. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

In diesem Thema wird beschrieben, wie Sie nur Spot-Instances in Ihrer Auto-Scaling-Gruppe starten können, indem Sie die Einstellungen in einer Startvorlage und nicht in der Auto-Scaling-Gruppe selbst festlegen. Die Informationen in diesem Thema gelten auch für Auto-Scaling-Gruppen, die Spot-Instances mit einer [Startkonfiguration](#) anfordern. Der Unterschied besteht darin, dass für eine Startkonfiguration ein Höchstpreis erforderlich ist, bei Startvorlagen ist der Höchstpreis jedoch optional.

Wenn Sie eine Startvorlage erstellen, um nur Spot-Instanzen zu starten, sollten Sie die folgenden Punkte beachten:

- **Spot-Preis.** Sie zahlen nur den aktuellen Spot-Preis für die Spot-Instances, die Sie starten. Dieser Preis ändert sich im Laufe der Zeit langsam, basierend auf den langfristigen Trends bei Angebot und Nachfrage. Weitere Informationen finden Sie unter [Spot-Instances](#) und [Preise und Einsparungen](#) im Amazon EC2 EC2-Benutzerhandbuch.
- **Festlegen des Höchstpreises.** Sie können optional einen Höchstpreis pro Stunde für Spot-Instances in Ihre Startvorlage aufnehmen. Wenn Ihr Maximalpreis den aktuellen Spot-Preis übersteigt, erfüllt der Amazon-EC2-Spot-Service Ihre Anfrage sofort, sofern Kapazität verfügbar ist. Wenn der Preis für Spot-Instances Ihren Maximalpreis für eine laufende Instance in Ihrer Auto-Scaling-Gruppe übersteigt, wird Ihre Instance beendet.

 Warning

Ihre Anwendung läuft möglicherweise nicht, wenn Sie keine Spot-Instances erhalten, z. B. wenn Ihr Höchstpreis zu niedrig ist. Um so lange wie möglich von den verfügbaren Spot-Instances zu profitieren, legen Sie Ihren Maximalpreis nahe dem On-Demand-Preis fest.

- **Ausgleichen zwischen den Availability Zones.** Wenn Sie mehrere Availability Zones angeben, verteilt Amazon EC2 Auto Scaling die Spot-Anfragen über die angegebenen Zonen. Ist Ihr Höchstpreis in einer Availability Zone zu niedrig, um Anfragen zu erfüllen, prüft Amazon-EC2-Auto-Scaling, ob Anfragen in anderen Availability Zones erfolgreich waren. Ist dies der Fall, beendet Amazon EC2 Auto Scaling die nicht erfolgreichen Anfragen und verteilt sie auf die Availability Zones, deren Anfragen erfolgreich waren. Fallen die Preise in einer Availability Zone ohne erfolgreiche Anfragen so weit, dass künftige Anfragen erfolgreich sind, gleicht Amazon EC2 Auto Scaling die Kapazitäten der Availability Zones wieder aus.
- **Spot-Instance-Beendigung.** Spot-Instances können jederzeit gekündigt werden. Der Amazon-EC2-Spot-Service kann Spot-Instances in Ihrer Auto-Scaling-Gruppe beenden, wenn sich die Verfügbarkeit von Spot-Instances oder der Preis für diese ändert. Bei der Skalierung oder bei der Durchführung von Gesundheitsprüfungen kann Amazon EC2 Auto Scaling Spot-Instances auf die gleiche Weise beenden wie On-Demand-Instances. Wenn eine Instance beendet wird, wird jeglicher Speicher gelöscht.
- **Behalten Sie Ihre gewünschte Kapazität bei.** Wenn eine Spot-Instance beendet wird, versucht Amazon EC2 Auto Scaling, eine andere Spot-Instance zu starten, um die gewünschte Kapazität für die Gruppe aufrechtzuerhalten. Wenn der aktuelle Spot-Preis unter Ihrem Höchstpreis liegt, wird eine Spot-Instance gestartet. Wenn die Anfrage nach einer Spot-Instance erfolglos ist, versucht sie es weiter.

- **Ändern des Höchstpreises.** Um Ihren Höchstpreis zu ändern, erstellen Sie eine neue Startvorlage oder aktualisieren Sie eine vorhandene Startvorlage mit dem neuen Höchstpreis und verknüpfen Sie sie dann mit Ihrer Auto-Scaling-Gruppe. Die bestehenden Spot-Instances laufen weiter, solange der in der für diese Instances verwendeten Startvorlage angegebene Höchstpreis höher ist als der aktuelle Spot-Preis. Wenn Sie keinen Höchstpreis festgelegt haben, ist der Standardhöchstpreis der Preis auf Abruf.

Capacity Blocks für Machine-Learning-Workloads verwenden

Capacity Blocks helfen Ihnen dabei, stark nachgefragte GPU-Instances zu einem future Zeitpunkt zu reservieren, um Ihre kurzfristigen Machine-Learning-Workloads (ML) zu unterstützen.

Einen Überblick über Capacity Blocks und wie sie funktionieren, finden Sie unter [Capacity Blocks für ML](#) im Amazon EC2 EC2-Benutzerhandbuch.

Um mit der Nutzung zu beginnen Capacity Blocks, erstellen Sie eine Kapazitätsreservierung in einer bestimmten Availability Zone. Capacity Blocks werden als `targeted` Kapazitätsreservierungen in einer einzigen Availability Zone bereitgestellt. Wenn Sie Ihre Startvorlage erstellen, geben Sie die Reservierungs-ID und den Instanztyp des Kapazitätsblocks an. Aktualisieren Sie dann Ihre Auto Scaling Group so, dass sie die von Ihnen erstellte Startvorlage und die Availability Zone des Capacity Blocks verwendet. Wenn Ihre Capacity Block-Reservierung beginnt, verwenden Sie die geplante Skalierung, um dieselbe Anzahl von Instances wie Ihre Capacity Block-Reservierung zu starten.

Important

Capacity Blocks sind nur für bestimmte Amazon EC2 Instance-Typen und AWS-Regionen verfügbar. Weitere Informationen finden Sie unter [Voraussetzungen](#) im Amazon EC2-Benutzerhandbuch.

Inhalt

- [Betriebliche Richtlinien](#)
- [Geben Sie in Ihrer Startvorlage einen Kapazitätsblock an](#)
- [Einschränkungen](#)
- [Zugehörige Ressourcen](#)

Betriebliche Richtlinien

Nachfolgend finden Sie grundlegende Richtlinien, die Sie bei der Verwendung eines Kapazitätsblocks mit einer Auto-Scaling-Gruppe beachten sollten.

- Skalieren Sie Ihre Auto-Scaling-Gruppe mehr als 30 Minuten vor der Endzeit der Kapazitätsblockreservierung auf Null herunter. Amazon EC2 beendet alle Instances, die noch in Betrieb sind, 30 Minuten vor dem Ende des Kapazitätsblocks.
- Wir empfehlen Ihnen, die geplante Skalierung zu verwenden, um zu den entsprechenden Reservierungszeiten die horizontale Skalierung (Hinzufügen von Instances) und die Skalierung (Instances entfernen) durchzuführen. Weitere Informationen finden Sie unter [Geplante Skalierung für Amazon EC2 Auto Scaling](#).
- Fügen Sie bei Bedarf Lebenszyklus-Hooks hinzu, um Ihre Anwendung beim Skalieren innerhalb der Instances ordnungsgemäß herunterzufahren. Lassen Sie genügend Zeit, bis die Lebenszyklus-Aktion abgeschlossen ist, bevor Amazon EC2 beginnt, Ihre Instances 30 Minuten vor dem Ende der Kapazitätsblockreservierung zwangsweise zu beenden. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).
- Stellen Sie sicher, dass die Auto-Scaling-Gruppe für die gesamte Dauer der Reservierung auf die richtige Version der Startvorlage verweist. Wir empfehlen, auf eine bestimmte Version der Startvorlage statt auf die Version `$Default` oder `$Latest` zu verweisen.

Note

Wenn Sie eine Capacity Block-Instance bis zum Ende der Reservierung laufen lassen und Amazon EC2 sie zurückfordert, geben die Skalierungsaktivitäten für Ihre Auto Scaling Scaling-Gruppe an, dass sie "taken out of service in response to an EC2 health check that indicated it had been terminated or stopped", war, obwohl sie am Ende des Kapazitätsblocks absichtlich zurückgefordert wurde. In ähnlicher Weise versucht Amazon EC2 Auto Scaling, die Instance auf dieselbe Weise zu ersetzen, wie es bei jeder Instance der Fall ist, die eine Zustandsprüfung nicht besteht. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Geben Sie in Ihrer Startvorlage einen Kapazitätsblock an

Verwenden Sie eine der folgenden Methoden, um eine Startvorlage zu erstellen, die auf einen bestimmten Kapazitätsblock für Ihre Auto Scaling Scaling-Gruppe abzielt:

Console

Angabe eines Kapazitätsblocks in Ihrer Startvorlage (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie in der oberen Navigationsleiste den Ort aus, AWS-Region an dem Sie Ihren Kapazitätsblock erstellt haben.
3. Wählen Sie im Navigationsbereich unter Instances die Option Launch Templates aus.
4. Wählen Sie Startvorlage erstellen und erstellen Sie die Startvorlage. Schließen Sie bei Bedarf die ID des Amazon Machine Image (AMI), den Instance-Typ und alle anderen Startvorlagen ein.
5. Erweitern Sie den Abschnitt Erweiterte Details, um die erweiterten Einstellungen anzuzeigen.
6. Wählen Sie als Kaufoption Kapazitätsblöcke aus.
7. Wählen Sie für Kapazitätsreservierung die Option Ziel nach ID und dann für Kapazitätsreservierung – Ziel nach ID die Kapazitätsreservierungs-ID eines vorhandenen Kapazitätsblocks aus.
8. Klicken Sie danach auf Startvorlage erstellen.

Hilfe zum Erstellen einer Auto Scaling Scaling-Gruppe mit einer Startvorlage finden Sie unter [Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage](#).

AWS CLI

Angabe eines Kapazitätsblocks in Ihrer Startvorlage (AWS CLI)

Verwenden Sie den folgenden [create-launch-template](#) Befehl, um eine Startvorlage zu erstellen, die eine vorhandene Kapazitätsblock-Reservierungs-ID angibt. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

```
aws ec2 create-launch-template --launch-template-name my-template-for-capacity-block \
  --version-description AutoScalingVersion1 --region us-east-2 \
  --launch-template-data file://config.json
```

i Tip

Wenn dieser Befehl einen Fehler auslöst, stellen Sie sicher, dass Sie den Befehl AWS CLI lokal auf die neueste Version aktualisiert haben.

Inhalt von `config.json`.

```
{
  "ImageId": "ami-04d5cc9b88example",
  "InstanceType": "p4d.24xlarge",
  "SecurityGroupIds": [
    "sg-903004f88example"
  ],
  "KeyName": "MyKeyPair",
  "InstanceMarketOptions": {
    "MarketType": "capacity-block"
  },
  "CapacityReservationSpecification": {
    "CapacityReservationTarget": {
      "CapacityReservationId": "cr-02168da1478b509e0"
    }
  }
}
```

Es folgt eine Beispielausgabe.

```
{
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-068f72b724example",
    "LaunchTemplateName": "my-template-for-capacity-block",
    "CreateTime": "2023-10-27T15:12:44.000Z",
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "DefaultVersionNumber": 1,
    "LatestVersionNumber": 1
  }
}
```

Sie können den folgenden [describe-launch-template-versions](#) Befehl verwenden, um die Reservierungs-ID für den Kapazitätsblock zu überprüfen, die der Startvorlage zugeordnet ist.

```
aws ec2 describe-launch-template-versions --launch-template-names my-template-for-capacity-block \  
--region us-east-2
```

Es folgt eine Beispielausgabe für eine Startvorlage mit Angabe einer Kapazitätsblockreservierung.

```
{  
  "LaunchTemplateVersions": [  
    {  
      "LaunchTemplateId": "lt-068f72b724example",  
      "LaunchTemplateName": "my-template-for-capacity-block",  
      "VersionNumber": 1,  
      "CreateTime": "2023-10-27T15:12:44.000Z",  
      "CreatedBy": "arn:aws:iam::123456789012:user/Bob",  
      "DefaultVersion": true,  
      "LaunchTemplateData": {  
        "ImageId": "ami-04d5cc9b88example",  
        "InstanceType": "p5.48xlarge",  
        "SecurityGroupIds": [  
          "sg-903004f88example"  
        ],  
        "KeyName": "MyKeyPair",  
        "InstanceMarketOptions": {  
          "MarketType": "capacity-block"  
        },  
        "CapacityReservationSpecification": {  
          "CapacityReservationTarget": {  
            "CapacityReservationId": "cr-02168da1478b509e0"  
          }  
        }  
      }  
    }  
  ]  
}
```

Einschränkungen

- Support für Capacity Blocks ist nur verfügbar, wenn Ihre Auto Scaling Scaling-Gruppe über eine kompatible Konfiguration verfügt. Gruppen mit gemischten Instances und warmen Pools werden nicht unterstützt.

- Sie können jeweils nur einen Kapazitätsblock als Ziel festlegen.

Zugehörige Ressourcen

- Die Voraussetzungen und Empfehlungen für die Verwendung von P5-Instances finden [Sie unter Erste Schritte mit P5-Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Amazon EKS unterstützt die Verwendung Capacity Blocks zur Unterstützung Ihrer kurzfristigen Workloads für maschinelles Lernen (ML) auf Amazon EKS-Clustern. Weitere Informationen finden Sie unter [Capacity Blocks für ML](#) im Amazon EKS-Benutzerhandbuch.
- Sie können es Capacity Blocks mit unterstützten Instance-Typen und Regionen verwenden. Kapazitätsreservierungen auf Abruf bieten jedoch die Flexibilität, Kapazität für andere Instance-Typen und Regionen zu reservieren. Ein Tutorial, das Ihnen zeigt, wie Sie die Option On-Demand-Kapazitätsreservierung verwenden, finden Sie unter [Reservieren Sie Kapazität in bestimmten Availability Zones mit Kapazitätsreservierungen](#).

Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten

Ab dem 1. Januar 2023 werden neue Instance-Typen in Startkonfigurationen nicht mehr unterstützt. Dies gilt für alle Instance-Typen, die AWS-Region nach dem ersten Start der Region zu oder hinzugefügt wurden. Darüber hinaus können Sie möglicherweise eine Startkonfiguration mit einem Instance-Typ erstellen, der in einer Region nicht mehr unterstützt wird. Weitere Informationen finden Sie unter [Auto Scaling Scaling-Startkonfigurationen](#).

Gehen Sie wie folgt vor, um Ihre Auto Scaling Scaling-Gruppen von Startkonfigurationen zu Startvorlagen zu migrieren.

Important

Stellen Sie sicher, dass Sie über die erforderlichen Berechtigungen verfügen, um mit Startvorlagen zu arbeiten. Weitere Informationen finden Sie unter [Berechtigungen für die Arbeit mit Startvorlagen](#).

Stellen Sie sicher, dass mit der Startkonfiguration verknüpfte Ressourcen, wie Sicherheitsgruppen und IAM-Instanzprofile, nicht versehentlich gelöscht werden.

Testen Sie den Rollback-Vorgang, um sicherzustellen, dass Änderungen sicher rückgängig gemacht werden können.

Schritt 1: Suchen Sie Auto-Scaling-Gruppen, die Startkonfigurationen verwenden

Um festzustellen, ob Sie Auto Scaling Scaling-Gruppen haben, die noch Startkonfigurationen verwenden, führen Sie den folgenden [describe-auto-scaling-groups](#) Befehl mit dem aus AWS CLI. Ersetzen Sie **REGION** durch Ihre AWS-Region.

```
aws autoscaling describe-auto-scaling-groups --region REGION \  
--query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

Es folgt eine Beispielausgabe.

```
[  
  {  
    "AutoScalingGroupName": "group-1",  
    "AutoScalingGroupARN": "arn",  
    "LaunchConfigurationName": "my-launch-config",  
    "MinSize": 1,  
    "MaxSize": 5,  
    "DesiredCapacity": 2,  
    "DefaultCooldown": 300,  
    "AvailabilityZones": [  
      "us-west-2a",  
      "us-west-2b",  
      "us-west-2c"  
    ],  
    "LoadBalancerNames": [],  
    "TargetGroupARNs": [],  
    "HealthCheckType": "EC2",  
    "HealthCheckGracePeriod": 300,  
    "Instances": [  
      {  
        "ProtectedFromScaleIn": false,  
        "AvailabilityZone": "us-west-2a",  
        "LaunchConfigurationName": "my-launch-config",  
        "InstanceId": "i-05b4f7d5be44822a6",  
        "InstanceType": "t3.micro",
```

```

        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
    },
    {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2b",
        "LaunchConfigurationName": "my-launch-config",
        "InstanceId": "i-0c20ac468fa3049e8",
        "InstanceType": "t3.micro",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
    }
],
"CreatedTime": "2023-03-09T22:15:11.611Z",
"SuspendedProcesses": [],
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
"EnabledMetrics": [],
"Tags": [
    {
        "ResourceId": "group-1",
        "ResourceType": "auto-scaling-group",
        "Key": "environment",
        "Value": "production",
        "PropagateAtLaunch": true
    }
],
"TerminationPolicies": [
    "Default"
],
"NewInstancesProtectedFromScaleIn": false,
"ServiceLinkedRoleARN": "arn",
    "TrafficSources": []
},
    ... additional groups ...
]

```

Führen Sie alternativ den folgenden Befehl aus, um alles außer den Auto-Scaling-Gruppe-Namen mit den Namen ihrer jeweiligen Startkonfigurationen und Tags in der Ausgabe zu entfernen:

```
aws autoscaling describe-auto-scaling-groups --region REGION \
```

```
--query 'AutoScalingGroups[?LaunchConfigurationName!=`null`].{AutoScalingGroupName: AutoScalingGroupName, LaunchConfigurationName: LaunchConfigurationName, Tags: Tags}'
```

Das folgende Beispiel zeigt eine Ausgabe.

```
[
  {
    "AutoScalingGroupName": "group-1",
    "LaunchConfigurationName": "my-launch-config",
    "Tags": [
      {
        "ResourceId": "group-1",
        "ResourceType": "auto-scaling-group",
        "Key": "environment",
        "Value": "production",
        "PropagateAtLaunch": true
      }
    ]
  },
  ... additional groups ...
]
```

Weitere Informationen zum Filtern finden Sie im AWS Command Line Interface Benutzerhandbuch unter [Filtern der AWS CLI Ausgabe](#).

Schritt 2: Kopieren einer Startkonfiguration in eine Startvorlage

Mit dem folgenden Verfahren können Sie eine Startkonfiguration in eine Startvorlage kopieren. Dann können Sie sie zu Ihrer Auto-Scaling-Gruppe hinzufügen.

Das Kopieren mehrerer Startkonfigurationen führt zu Startvorlagen mit identischem Namen. Um den Namen zu ändern, der einer Startvorlage während des Kopiervorgangs gegeben wurde, müssen Sie die Startkonfigurationen eine nach der anderen kopieren.

Note

Die Kopierfunktion steht nur über die Konsole zur Verfügung.

Kopieren einer Startkonfiguration in eine Startvorlage (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im linken Navigationsbereich unter Auto Scaling Auto-Scaling-Gruppen aus.
3. Wählen Sie oben auf der Seite Startkonfigurationen aus. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Startkonfigurationen anzeigen aus, um zu bestätigen, dass Sie die Seite Startkonfigurationen aufrufen möchten.
4. Wählen Sie die zu kopierende Startkonfiguration und Copy to launch template, Copy selected (In Startvorlage kopieren, Kopie ausgewählt) aus. Dadurch wird eine neue Startvorlage mit demselben Namen und denselben Optionen wie bei der ausgewählten Startkonfiguration eingerichtet.
5. Unter New launch template name (Neuer Startvorlagenname) können Sie den Namen der Startkonfiguration (Standard) verwenden oder einen neuen Namen eingeben. Die Namen von Startvorlagen müssen eindeutig sein.
6. (Optional) Wählen Sie Eine Auto-Scaling-Gruppe mithilfe der neuen Vorlage erstellen aus.

Sie können diesen Schritt überspringen, wenn Sie das Kopieren der Startkonfiguration abschließen möchten. Sie müssen keine neue Auto-Scaling-Gruppe erstellen.

7. Wählen Sie die Option Kopieren aus.

So kopieren Sie alle Startkonfigurationen in Startvorlagen (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich unter Auto Scaling die Option Launch Configurations (Startkonfigurationen) aus.
3. Klicken Sie auf Kopieren zur Startvorlage, Alle kopieren. Dadurch wird jede Startkonfiguration in der aktuellen Region in eine neue Startvorlage mit demselben Namen und denselben Optionen kopiert.
4. Wählen Sie die Option Kopieren aus.

Schritt 3: Aktualisieren einer Auto-Scaling-Gruppe zum Verwenden einer Startvorlage

Wenn Sie eine Startvorlage erstellt haben, können Sie sie zu Ihrer Auto-Scaling-Gruppe hinzufügen.

Aktualisieren einer Auto-Scaling-Gruppe zum Verwenden einer Startvorlage (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet, in dem Informationen über die ausgewählte Gruppe angezeigt werden.

3. Wählen Sie auf der Registerkarte Details die Option Konfiguration starten, Bearbeiten aus.
4. Wählen Sie So wechseln Sie zur Startvorlage aus.
5. Wählen Sie als Launch Template (Startvorlage) Ihre Startvorlage aus.
6. Als Version wählen Sie ggf. die Version der Startvorlage aus. Nachdem Sie Versionen einer Startvorlage erstellt haben, können Sie auswählen, ob die Auto-Scaling-Gruppe beim Hochskalieren die standardmäßige oder die neueste Version der Startvorlage verwenden soll.
7. Wählen Sie Aktualisieren aus.

Aktualisieren einer Auto-Scaling-Gruppe zum Verwenden einer Startvorlage (AWS CLI)

Mit dem folgenden [update-auto-scaling-group](#) Befehl wird die angegebene Auto Scaling Group aktualisiert, sodass sie die ursprüngliche Version der angegebenen Startvorlage verwendet.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
```

Weitere Beispiele für die Verwendung von CLI-Befehlen, um eine Auto-Scaling-Gruppe zur Verwendung einer Startvorlage zu aktualisieren, finden Sie unter [Aktualisieren einer Auto-Scaling-Gruppe zum Verwenden einer Startvorlage](#).

Schritt 4: Ersetzen Ihrer Instances

Wenn Sie die Startkonfiguration durch eine Startvorlage ersetzt haben, verwenden alle neuen Instances die neue Startvorlage. Bestehende Instances sind nicht betroffen.

Um vorhandene Instances zu aktualisieren, können Sie eine Instance-Aktualisierung verwenden, um die Instances in der Auto-Scaling-Gruppe zu ersetzen, anstatt Instances gleichzeitig manuell zu ersetzen. Weitere Informationen finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um](#)

[Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#). Eine Instance-Aktualisierung kann besonders hilfreich sein, wenn die Gruppe groß ist.

Alternativ können Sie die automatische Skalierung zulassen, um vorhandene Instances auf Grundlage der [Beendigungsrichtlinien](#) der Gruppe schrittweise durch neue Instances zu ersetzen, oder Sie können sie beenden. Das manuelle Beenden zwingt Ihre Auto-Scaling-Gruppe, neue Instances zu starten, um die gewünschte Kapazität der Gruppe aufrechtzuerhalten. Weitere Informationen finden Sie unter [Terminate an Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Zusätzliche Informationen

Weitere Informationen finden Sie im AWS Compute-Blog unter [Amazon EC2 Auto Scaling wird keine Unterstützung mehr für neue EC2-Funktionen zu Launch-Konfigurationen hinzufügen](#).

Ein Thema, das Ihnen zeigt, wie Sie AWS CloudFormation Stacks von Startkonfigurationen zu Startvorlagen migrieren, finden Sie unter [Migrieren Sie AWS CloudFormation Stacks zu Startvorlagen](#)

Migrieren Sie AWS CloudFormation Stacks zu Startvorlagen

Sie können Ihre vorhandenen CloudFormation Stack-Vorlagen von Startkonfigurationen zu Startvorlagen migrieren. Fügen Sie dazu eine Startvorlage direkt zu einer vorhandenen Stack-Vorlage hinzu und verknüpfen Sie die Startvorlage dann mit der Auto-Scaling-Gruppe in der Stack-Vorlage. Verwenden Sie anschließend die geänderte Vorlage zum Aktualisieren Ihres Stacks.

Bei der Migration zu Startvorlagen spart Ihnen dieses Thema Zeit, da es Anweisungen zum Umschreiben der Startkonfigurationen in Ihren CloudFormation Stack-Vorlagen als Startvorlagen enthält. Weitere Informationen zum Migrieren von Startkonfigurationen in Startvorlagen finden Sie unter [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).

Themen

- [Auto-Scaling-Gruppen finden, die eine Startkonfiguration verwenden](#)
- [Aktualisieren eines Stacks zur Verwendung einer Startvorlage](#)
- [Das Aktualisierungsverhalten von Stack-Ressourcen verstehen](#)
- [Verfolgen Sie die Migration](#)
- [Referenz für die Abbildung der Startkonfiguration](#)

Auto-Scaling-Gruppen finden, die eine Startkonfiguration verwenden

So finden Sie Auto-Scaling-Gruppen, die eine Startkonfiguration verwenden

- Verwenden Sie den folgenden [describe-auto-scaling-groups](#) Befehl, um die Namen der Auto Scaling Scaling-Gruppen aufzulisten, die Startkonfigurationen in der angegebenen Region verwenden. Fügen Sie die `--filters` Option hinzu, die Ergebnisse auf Gruppen einzugrenzen, die einem CloudFormation Stack zugeordnet sind (durch Filtern nach dem `aws:cloudformation:stack-name` Tag-Schlüssel).

```
aws autoscaling describe-auto-scaling-groups --region REGION \  
  --filters Name=tag-key,Values=aws:cloudformation:stack-name \  
  --query 'AutoScalingGroups[?LaunchConfigurationName!  
= `null` ].AutoScalingGroupName'
```

Das folgende Beispiel zeigt eine Ausgabe.

```
[  
  "{stack-name}-group-1",  
  "{stack-name}-group-2",  
  "{stack-name}-group-3"  
]
```

Sie finden weitere nützliche AWS CLI Befehle, um Auto Scaling Scaling-Gruppen für die Migration zu finden und die Ausgabe zu filtern [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).

Important

Wenn Ihre Stack-Ressourcen AWSEB in ihrem Namen stehen, bedeutet das, dass sie durch erstellt wurden AWS Elastic Beanstalk. In diesem Fall müssen Sie die Beanstalk-Umgebung aktualisieren, um Elastic Beanstalk anzuweisen, die Startkonfiguration zu entfernen und sie durch eine Startvorlage zu ersetzen.

Aktualisieren eines Stacks zur Verwendung einer Startvorlage

Befolgen Sie die Schritte in diesem Abschnitt, um Folgendes zu tun:

- Schreiben Sie die Startkonfiguration als Startvorlage um und verwenden Sie die entsprechenden Eigenschaften der Startvorlage.
- Verknüpfen Sie die neue Startvorlage mit der Auto-Scaling-Gruppe.
- Stellen Sie diese Updates bereit.

So ändern Sie die Stack-Vorlage und aktualisieren den Stack

1. Folgen Sie den gleichen allgemeinen Verfahren zum Ändern der Stack-Vorlage, die im AWS CloudFormation Benutzerhandbuch unter [Ändern einer Stack-Vorlage](#) beschrieben sind.
2. Schreiben Sie die Startkonfiguration in eine Startvorlage um. Sehen Sie sich das folgende Beispiel an:

Beispiel: Eine einfache Startkonfiguration

```
---
Resources:
  myLaunchConfig:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId: ami-02354e95b3example
      InstanceType: t3.micro
      SecurityGroups:
        - !Ref EC2SecurityGroup
      KeyName: MyKeyPair
      BlockDeviceMappings:
        - DeviceName: /dev/xvda
          Ebs:
            VolumeSize: 150
            DeleteOnTermination: true
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource myASG
          --region ${AWS::Region}
```

Beispiel: Das Äquivalent zur Startvorlage

```
---
Resources:
```

```

myLaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
    LaunchTemplateData:
      ImageId: ami-02354e95b3example
      InstanceType: t3.micro
      SecurityGroupIds:
        - Ref! EC2SecurityGroup
      KeyName: MyKeyPair
      BlockDeviceMappings:
        - DeviceName: /dev/xvda
          Ebs:
            VolumeSize: 150
            DeleteOnTermination: true
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -x
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
myASG --region ${AWS::Region}

```

Referenzinformationen zu allen Eigenschaften, die Amazon EC2 unterstützt, finden Sie [AWS::EC2::LaunchTemplate](#) im AWS CloudFormation Benutzerhandbuch.

Beachten Sie, dass die Startvorlage die Eigenschaft `LaunchTemplateName` mit einem Wert von `!Sub ${AWS::StackName}-launch-template` enthält. Dies ist erforderlich, wenn der Name der Startvorlage den Namen des Stacks enthalten soll.

3. Wenn die Eigenschaft **IamInstanceProfile** in Ihrer Startkonfiguration vorhanden ist, müssen Sie sie in eine Struktur umwandeln und entweder den Namen oder den ARN des Instance-Profils angeben. Ein Beispiel finden Sie unter [AWS::EC2::LaunchTemplate](#).
4. Wenn die Eigenschaften **AssociatePublicIpAddress**, **InstanceMonitoring** oder **PlacementTenancy** in Ihrer Startkonfiguration vorhanden sind, müssen Sie diese in eine Struktur umwandeln. Beispiele finden Sie unter [AWS::EC2::LaunchTemplate](#).

Eine Ausnahme besteht, wenn der Wert für die Eigenschaft `MapPublicIpOnLaunch` in den Teilnetzen, die Sie für Ihre Auto-Scaling-Gruppe verwendet haben, mit dem Wert für die Eigenschaft `AssociatePublicIpAddress` in Ihrer Startkonfiguration übereinstimmt. In diesem Fall können Sie die `AssociatePublicIpAddress`-Eigenschaft ignorieren. Die `AssociatePublicIpAddress` Eigenschaft wird nur verwendet, um die

MapPublicIpOnLaunch Eigenschaft zu überschreiben, um zu ändern, ob Instances beim Start eine öffentliche IPv4 Adresse erhalten.

5. Sie können Sicherheitsgruppen aus der **SecurityGroups**-Eigenschaft an eine von zwei Stellen in Ihrer Startvorlage kopieren. Normalerweise kopieren Sie die Sicherheitsgruppen in die SecurityGroupIds-Eigenschaft. Wenn Sie jedoch in Ihrer Startvorlage eine NetworkInterfaces-Struktur erstellen, um die AssociatePublicIpAddress-Eigenschaft anzugeben, müssen Sie stattdessen die Sicherheitsgruppen in die Groups-Eigenschaft der Netzwerkschnittstelle kopieren.
6. Wenn in Ihrer Startkonfiguration BlockDeviceMapping-Strukturen vorhanden sind und **NoDevice** auf true gesetzt ist, müssen Sie in Ihrer Startvorlage eine leere Zeichenfolge für NoDevice angeben, damit Amazon EC2 das Gerät auslässt.
7. Wenn die Eigenschaft **SpotPrice** in Ihrer Startkonfiguration vorhanden ist, empfehlen wir Ihnen, sie in Ihrer Startvorlage wegzulassen. Ihre Spot Instance wird zum aktuellen Spot-Preis gestartet. Dieser Preis wird niemals den On-Demand-Preis überschreiten.

Um Spot-Instances anzufordern, haben Sie zwei Optionen, die sich gegenseitig ausschließen:

- Die erste Möglichkeit besteht darin, die InstanceMarketOptions-Struktur in Ihrer Startvorlage zu verwenden (nicht empfohlen). Weitere Informationen finden Sie [AWS::EC2::LaunchTemplate InstanceMarketOptions](#) im AWS CloudFormation Benutzerhandbuch.
 - Die andere besteht darin, Ihrer Auto-Scaling-Gruppe eine MixedInstancesPolicy-Struktur hinzuzufügen. Auf diese Weise stehen Ihnen mehr Optionen zur Verfügung, wie Sie die Anfrage stellen können. Eine Spot-Instance-Anfrage in Ihrer Startvorlage unterstützt nicht mehr als eine Instance-Typauswahl pro Auto-Scaling-Gruppe. Eine Richtlinie für gemischte Instances unterstützt jedoch die Auswahl von mehr als einem Instance-Typ pro Auto-Scaling-Gruppe. Spot-Instance-Anfragen profitieren davon, dass mehr als ein einziger Instance-Typ zur Auswahl steht. Weitere Informationen finden Sie MixedInstancesPolicy im AWS CloudFormation Benutzerhandbuch unter [AWS::AutoScaling::AutoScaling MixedInstancesPolicy](#).
8. Entfernen Sie die **LaunchConfigurationName** Eigenschaft aus der [AWS::AutoScaling::AutoScaling](#). Fügen Sie stattdessen die Startvorlage hinzu.

In den folgenden Beispielen ruft die intrinsische Funktion [Ref](#) die ID der [AWS::EC2::LaunchTemplate](#) Ressource mit der logischen ID ab. myLaunchTemplate Die

[GetAtt](#)Funktion ruft die neueste Versionsnummer (z. B.1) der Startvorlage für die `Version` Eigenschaft ab.

Beispiel: Ohne eine Richtlinie für gemischte Instances

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref myLaunchTemplate
        Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    ...
```

Beispiel: Mit einer Richtlinie für gemischte Instances

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref myLaunchTemplate
            Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    ...
```

Referenzinformationen zu allen Eigenschaften, die Amazon EC2 Auto Scaling unterstützt, finden Sie unter [AWS::AutoScaling::AutoScalingGroup](#) im AWS CloudFormation Benutzerhandbuch.

9. Wenn Sie bereit sind, diese Updates bereitzustellen, folgen Sie den CloudFormation Verfahren, um den Stack mit Ihrer geänderten Stack-Vorlage zu aktualisieren. Weitere Informationen finden Sie unter [Ändern einer Stack-Vorlage](#) im AWS CloudFormation Benutzerhandbuch.

Das Aktualisierungsverhalten von Stack-Ressourcen verstehen

CloudFormation aktualisiert die Stack-Ressourcen, indem die Änderungen zwischen der von Ihnen bereitgestellten aktualisierten Vorlage und den Ressourcenkonfigurationen, die Sie in der vorherigen Version Ihrer Stack-Vorlage beschrieben haben, verglichen werden. Nicht geänderte Ressourcenkonfigurationen bleiben während des Updates davon unberührt.

CloudFormation unterstützt das [UpdatePolicy](#) Attribut für Auto Scaling Scaling-Gruppen. Wenn während eines Updates auf eingestellt `UpdatePolicy` ist `AutoScalingRollingUpdate`, werden `InService` Instanzen CloudFormation ersetzt, nachdem Sie die Schritte in diesem Verfahren ausgeführt haben. Wenn auf gesetzt `UpdatePolicy` ist `AutoScalingReplacingUpdate`, CloudFormation ersetzt die Auto Scaling Scaling-Gruppe und ihren Warmpool (falls vorhanden).

Wenn Sie kein `UpdatePolicy` Attribut für Ihre Auto Scaling Scaling-Gruppe angegeben haben, wird die Startvorlage auf ihre Richtigkeit überprüft, aber CloudFormation es werden keine Änderungen für die Instances in der Auto Scaling Scaling-Gruppe bereitgestellt. Alle neuen Instances verwenden Ihre Startvorlage, aber bestehende Instances werden weiterhin mit der Startkonfiguration ausgeführt, mit der sie ursprünglich gestartet wurden (obwohl die Startkonfiguration nicht mehr existiert). Die Ausnahme ist, wenn Sie Ihre Kaufoptionen ändern, z. B. indem Sie eine Police für gemischte Instances hinzufügen. In diesem Fall ersetzt Ihre Auto-Scaling-Gruppe die vorhandenen Instances nach und nach durch neue Instances, die den neuen Kaufoptionen entsprechen.

Wenn Sie eine Änderung rückgängig machen müssen, um von Startkonfigurationen zu Startvorlagen zu wechseln, stellen Sie sicher, dass Sie den Rollback-Vorgang testen.

Verfolgen Sie die Migration

So verfolgen Sie die Migration

1. Wählen Sie in der [CloudFormation -Konsole](#) den Stack aus, den Sie aktualisiert haben, und klicken Sie dann auf die Registerkarte Events (Ereignisse), um die Stack-Ereignisse anzuzeigen.
2. Um die Ereignisliste mit den neuesten Ereignissen zu aktualisieren, klicken Sie in der CloudFormation Konsole auf die Schaltfläche „Aktualisieren“.
3. Während Ihr Stack aktualisiert wird, werden Sie mehrere Ereignisse für jede Ressourcenaktualisierung feststellen. Wenn Sie in der Spalte Statusgrund eine Ausnahme sehen, die auf ein Problem beim Erstellen der Startvorlage hinweist, finden Sie weitere Informationen [Fehlersuche bei Amazon EC2 Auto Scaling: Startvorlagen](#) zu möglichen Ursachen.

4. (Optional) Je nach Verwendung des UpdatePolicy-Attributs können Sie den Fortschritt Ihrer Auto-Scaling-Gruppe auf der Seite [Auto-Scaling-Gruppen](#) der Amazon EC2-Konsole überwachen. Wählen Sie die Auto-Scaling-Gruppe aus. Auf der Registerkarte Activity (Aktivität) unter Activity history (Aktivitätsverlauf) zeigt die Spalte Status an, ob Ihre Auto-Scaling-Gruppe-Instances erfolgreich gestartet oder beendet hat oder ob die Skalierungsaktivität noch im Gange ist.
5. Wenn das Stack-Update abgeschlossen ist, wird CloudFormation ein UPDATE_COMPLETE Stack-Ereignis ausgelöst. Weitere Informationen finden Sie unter [Überwachung des Fortschritts einer Stack-Aktualisierung](#) im AWS CloudFormation Benutzerhandbuch.
6. Nachdem das Stack-Update abgeschlossen ist, öffnen [Sie die Seite Startvorlagen](#) und [Startkonfigurationen](#) der Amazon EC2-Konsole. Sie werden feststellen, dass eine neue Startvorlage erstellt und die Startkonfiguration gelöscht wurde.

Referenz für die Abbildung der Startkonfiguration

Zu Referenzzwecken sind in der folgenden Tabelle alle Eigenschaften der [AWS::AutoScaling::LaunchConfiguration](#) Ressource auf oberster Ebene mit ihren entsprechenden Eigenschaften in der [AWS::EC2::LaunchTemplate](#) Ressource aufgeführt.

Startkonfiguration Quelleigenschaft	Ziel-Eigenschaft der Startvorlage
AssociatePublicIpAddress	NetworkInterfaces.AssociatePublicIpAddress
BlockDeviceMappings	BlockDeviceMappings
ClassicLinkVPCId	Nicht verfügbar ¹
ClassicLinkVPCSecurityGroups	Nicht verfügbar ¹
EbsOptimized	EbsOptimized
IamInstanceProfile	Entweder IamInstanceProfile.Arn oder IamInstanceProfile.Name , jedoch nicht beides.
ImageId	ImageId

Startkonfiguration Quelleigenschaft	Ziel-Eigenschaft der Startvorlage
InstanceId	InstanceId
InstanceMonitoring	Monitoring.Enabled
InstanceType	InstanceType
KernelId	KernelId
KeyName	KeyName
LaunchConfigurationName	LaunchTemplateName
MetadataOptions	MetadataOptions
PlacementTenancy	Placement.Tenancy
RamDiskId	RamDiskId
SecurityGroups	Entweder SecurityGroupIds oder NetworkInterfaces.Groups , jedoch nicht beides.
SpotPrice	InstanceMarketOptions.SpotOptions.MaxPrice
UserData	UserData

¹ Die ClassicLinkVPCSecurityGroups Eigenschaften ClassicLinkVPCId und können nicht in einer Startvorlage verwendet werden, da EC2-Classic nicht mehr verfügbar ist.

Beispiele für die Erstellung und Verwaltung von Startvorlagen mit dem AWS CLI

Sie können Startvorlagen mit den SDKs AWS-Managementkonsole, AWS Command Line Interface (AWS CLI) oder verwalten. In diesem Abschnitt finden Sie Beispiele für die Erstellung und Verwaltung von Startvorlagen für Amazon EC2 Auto Scaling aus dem AWS CLI.

Inhalt

- [Beispielverwendung](#)
- [Erstellen einer grundlegenden Startvorlage](#)
- [Angaben von Tags, die Instances beim Start kennzeichnen](#)
- [Angaben einer IAM-Rolle, die an Instances übergeben wird](#)
- [Zuweisen einer öffentlichen IP-Adresse](#)
- [Angaben eines Benutzerdatenskripts, das Instances beim Start konfiguriert](#)
- [Angaben einer Blockgerät-Zuweisung für ein AMI](#)
- [Festlegen von Dedicated Hosts zur Bereitstellung von Softwarelizenzen externer Anbieter](#)
- [Angaben einer vorhandenen Netzwerkschnittstelle](#)
- [Erstellen mehrerer Netzwerkschnittstellen](#)
- [Verwalten Ihrer Startvorlagen](#)
- [Aktualisieren einer Auto-Scaling-Gruppe zum Verwenden einer Startvorlage](#)

Beispielverwendung

```
{
  "LaunchTemplateName": "my-template-for-auto-scaling",
  "VersionDescription": "test description",
  "LaunchTemplateData": {
    "ImageId": "ami-04d5cc9b88example",
    "InstanceType": "t2.micro",
    "SecurityGroupIds": [
      "sg-903004f88example"
    ],
    "KeyName": "MyKeyPair",
    "Monitoring": {
      "Enabled": true
    },
    "Placement": {
      "Tenancy": "dedicated"
    },
    "CreditSpecification": {
      "CpuCredits": "unlimited"
    },
    "MetadataOptions": {
      "HttpTokens": "required",
```

```
        "HttpPutResponseHopLimit": 1,  
        "HttpEndpoint": "enabled"  
    }  
}  
}
```

Erstellen einer grundlegenden Startvorlage

Um eine grundlegende Startvorlage zu erstellen, verwenden Sie den [create-launch-template](#) Befehl wie folgt mit den folgenden Änderungen:

- Ersetzen Sie `ami-04d5cc9b88example` mit der ID des AMI, von dem aus die Instances gestartet werden sollen.
- Ersetzen Sie `t2.micro` mit einem Instance-Typ, der kompatibel mit dem angegebenen AMI ist.

In diesem Beispiel wird eine Startvorlage mit dem Namen `my-template-for-auto-scaling` erstellt. Wenn die mit dieser Einführungsvorlage erstellten Instances in einer Standard-VPC gestartet werden, erhalten sie standardmäßig eine öffentliche IP-Adresse. Wenn die Instances in einer nicht standardmäßigen VPC gestartet werden, erhalten sie keine öffentliche Adresse.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
  --launch-template-data  
  '{"ImageId":"ami-04d5cc9b88example", "InstanceType":"t2.micro"}'
```

Weitere Informationen zum Ansetzen von JSON-formatierten Parametern finden Sie unter [Verwenden von Anführungszeichen mit Zeichenfolgen in der AWS CLI](#) im AWS Command Line Interface -Benutzerhandbuch.

Alternativ können Sie die JSON-formatierten Parameter in einer Konfigurationsdatei angeben.

Im folgenden Beispiel wird eine einfache Startvorlage erstellt, die auf eine Konfigurationsdatei für Startvorlagenparameterwerte verweist.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
  --launch-template-data file://config.json
```

Inhalt von `config.json`:

```
{
  "ImageId":"ami-04d5cc9b88example",
  "InstanceType":"t2.micro"
}
```

Angeben von Tags, die Instances beim Start kennzeichnen

Im folgenden Beispiel wird Instances beim Start ein Tag hinzugefügt (z. B. purpose=webserver).

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"TagSpecifications":[{"ResourceType":"instance","Tags":
[{"Key":"purpose","Value":"webserver"}]}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.
```

Note

Wenn Sie Instance-Tags in Ihrer Startvorlage angeben und sich dann dafür entschieden haben, die Tags Ihrer Auto-Scaling-Gruppe an ihre Instances zu übertragen, werden alle Tags zusammengeführt. Wenn derselbe Tag-Schlüssel für einen Tag in Ihrer Startvorlage und einen Tag in Ihrer Auto-Scaling-Gruppe angegeben wird, hat der Tag-Wert aus der Gruppe Vorrang.

Angeben einer IAM-Rolle, die an Instances übergeben wird

Im folgenden Beispiel wird der Name des Instance-Profils angegeben, das der IAM-Rolle zugeordnet ist, das beim Start an Instances übergeben wird. Weitere Informationen finden Sie unter [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#).

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"IamInstanceProfile":{"Name":"my-instance-
profile"},"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

Zuweisen einer öffentlichen IP-Adresse

Im folgenden [create-launch-template](#) Beispiel wird die Startvorlage so konfiguriert, dass Instances, die in einer nicht standardmäßigen VPC gestartet wurden, öffentliche Adressen zugewiesen werden.

Note

Wenn Sie eine Netzwerkschnittstelle angeben, geben Sie einen Wert für Groups an, der den Sicherheitsgruppen der VPC, in der Ihre Auto-Scaling-Gruppe Instances starten wird, entspricht. Geben Sie die VPC-Subnetze als Eigenschaften der Auto-Scaling-Gruppe an.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true}]',"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

Angeben eines Benutzerdatenskripts, das Instances beim Start konfiguriert

Im folgenden Beispiel wird ein Benutzerdatenskript als base64-kodierte Zeichenfolge angegeben, die Instances beim Start konfiguriert. Für den [create-launch-template](#) Befehl sind Base64-kodierte Benutzerdaten erforderlich.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data
 '{"UserData":"IyEvYmLuL2Jhc...", "ImageId":"ami-04d5cc9b88example", "InstanceType":"t2.micro"}'
```

Angeben einer Blockgerät-Zuweisung für ein AMI

Im folgenden [create-launch-template](#) Beispiel wird eine Startvorlage mit einer Blockgerätezuweisung erstellt: einem 22-Gigabyte-EBS-Volume, dem die Zuordnung zugewiesen ist. /dev/xvdcz Das Volume /dev/xvdcz verwendet den Volume-Typ General Purpose SSD (gp2) und wird beim Beenden der Instance, an die es angehängt ist, gelöscht.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"BlockDeviceMappings":[{"DeviceName":"/dev/xvdcz","Ebs":
{"VolumeSize":22,"VolumeType":"gp2","DeleteOnTermination":true}]},'ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

Festlegen von Dedicated Hosts zur Bereitstellung von Softwarelizenzen externer Anbieter

Wenn Sie eine Host-Tenancy angeben, können Sie eine Host-Ressourcengruppe und eine License Manager-Konfiguration angeben, um berechtigte Softwarelizenzen von externen Anbietern bereitzustellen. Anschließend können Sie die Lizenzen mit dem folgenden Befehl auf EC2-Instances verwenden. [create-launch-template](#)

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"Placement":
{"Tenancy":"host","HostResourceGroupArn":"arn"}, "LicenseSpecifications":
[{"LicenseConfigurationArn":"arn"}], "ImageId":"ami-04d5cc9b88example", "InstanceType":"t2.micro"
```

Angaben einer vorhandenen Netzwerkschnittstelle

Im folgenden [create-launch-template](#)Beispiel wird die primäre Netzwerkschnittstelle so konfiguriert, dass sie eine vorhandene Netzwerkschnittstelle verwendet.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0, "NetworkInterfaceId":"eni-
b9a5ac93", "DeleteOnTermination":false]}, "ImageId":"ami-04d5cc9b88example", "InstanceType":"t2.mi
```

Erstellen mehrerer Netzwerkschnittstellen

Im folgenden [create-launch-template](#)Beispiel wird eine sekundäre Netzwerkschnittstelle hinzugefügt. Die primäre Netzwerkschnittstelle hat einen Geräteindex von 0, und die sekundäre Netzwerkschnittstelle hat einen Geräteindex von 1.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":[{"DeviceIndex":0, "Groups":
["sg-903004f88example"], "DeleteOnTermination":true}, {"DeviceIndex":1, "Groups":
["sg-903004f88example"], "DeleteOnTermination":true]}, "ImageId":"ami-04d5cc9b88example", "InstanceType":"t2.mi
```

Wenn Sie einen Instance-Typ verwenden, der mehrere Netzwerkkarten und Elastic Fabric-Adapter (EFAs) unterstützt, können Sie mit dem folgenden [create-launch-template](#)Befehl eine

sekundäre Schnittstelle zu einer sekundären Netzwerkkarte hinzufügen und EFA aktivieren. Weitere Informationen finden [Sie unter Hinzufügen einer EFA zu einer Startvorlage](#) im Amazon EC2 EC2-Benutzerhandbuch.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"NetworkCardIndex":0,"DeviceIndex":0,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true},
{"NetworkCardIndex":1,"DeviceIndex":1,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true}]',"ImageId":"ami-09d95
```

Warning

Der Instance-Typ p4d.24xlarge verursacht höhere Kosten als die anderen Beispiele in diesem Abschnitt. Weitere Informationen zu Preisen für P4d-Instances erhalten Sie unter [Preise für Amazon-EC2-P4d-Instances](#).

Note

Das Anfügen mehrerer Netzwerkschnittstellen aus demselben Subnetz an eine Instance kann asymmetrisches Routing einführen, insbesondere bei Instances, die eine Linux-Variante verwenden, die nicht von Amazon stammt. Wenn Sie diese Art von Konfiguration benötigen, müssen Sie die sekundäre Netzwerkschnittstelle innerhalb des Betriebssystems konfigurieren. Ein Beispiel finden Sie unter [Wie kann ich dafür sorgen, dass meine sekundäre Netzwerkschnittstelle in meiner Ubuntu EC2-Instance funktioniert?](#) im AWS Knowledge Center.

Verwalten Ihrer Startvorlagen

Das AWS CLI beinhaltet mehrere andere Befehle, mit denen Sie Ihre Startvorlagen verwalten können.

Inhalt

- [Auflisten und Beschreiben Ihrer Startvorlagen](#)
- [Erstellen einer Startvorlagenversion](#)

- [Löschen einer Startvorlagenversion](#)
- [Löschen einer Startvorlage](#)

Auflisten und Beschreiben Ihrer Startvorlagen

Sie können zwei AWS CLI Befehle verwenden, um Informationen zu Ihren Startvorlagen abzurufen: [describe-launch-templates](#) und [describe-launch-template-versions](#).

Mit dem [describe-launch-templates](#) Befehl können Sie eine Liste aller Startvorlagen abrufen, die Sie erstellt haben. Sie können eine Option verwenden, um Ergebnisse nach Namen einer Startvorlage, Zeit, Tag-Schlüssel oder einer Kombination aus Tag und Schlüssel-Wert filtern. Mit diesem Befehl werden zusammenfassende Informationen zu allen Ihren Startvorlagen zurückgegeben, einschließlich der Startvorlagenkennung, der neuesten Version und der Standardversion.

Das folgende Beispiel bietet eine Zusammenfassung der angegebenen Startvorlage.

```
aws ec2 describe-launch-templates --launch-template-names my-template-for-auto-scaling
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "LaunchTemplates": [
    {
      "LaunchTemplateId": "lt-068f72b729example",
      "LaunchTemplateName": "my-template-for-auto-scaling",
      "CreateTime": "2020-02-28T19:52:27.000Z",
      "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
      "DefaultVersionNumber": 1,
      "LatestVersionNumber": 1
    }
  ]
}
```

Wenn Sie nicht die `--launch-template-names`-Option verwenden, um die Ausgabe auf eine Startvorlage zu beschränken, werden Informationen zu allen Ihren Startvorlagen zurückgegeben.

Der folgende [describe-launch-template-versions](#) Befehl enthält Informationen zur Beschreibung der Versionen der angegebenen Startvorlage.

```
aws ec2 describe-launch-template-versions --launch-template-id lt-068f72b729example
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "LaunchTemplateVersions": [
    {
      "VersionDescription": "version1",
      "LaunchTemplateId": "lt-068f72b729example",
      "LaunchTemplateName": "my-template-for-auto-scaling",
      "VersionNumber": 1,
      "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
      "LaunchTemplateData": {
        "TagSpecifications": [
          {
            "ResourceType": "instance",
            "Tags": [
              {
                "Key": "purpose",
                "Value": "webserver"
              }
            ]
          }
        ],
        "ImageId": "ami-04d5cc9b88example",
        "InstanceType": "t2.micro",
        "NetworkInterfaces": [
          {
            "DeviceIndex": 0,
            "DeleteOnTermination": true,
            "Groups": [
              "sg-903004f88example"
            ],
            "AssociatePublicIpAddress": true
          }
        ],
        "DefaultVersion": true,
        "CreateTime": "2020-02-28T19:52:27.000Z"
      }
    ]
  ]
}
```

Erstellen einer Startvorlagenversion

Der folgende [create-launch-template-version](#) Befehl erstellt eine neue Version der Startvorlage, die auf Version 1 der Startvorlage basiert, und gibt eine andere AMI-ID an.

```
aws ec2 create-launch-template-version --launch-template-id lt-068f72b729example --  
version-description version2 \  
--source-version 1 --launch-template-data "ImageId=ami-c998b6b2example"
```

Verwenden Sie den [modify-launch-template](#) Befehl, um die Standardversion der Startvorlage festzulegen.

Löschen einer Startvorlagenversion

Der folgende [delete-launch-template-versions](#) Befehl löscht die angegebene Version der Startvorlage.

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b729example --  
versions 1
```

Löschen einer Startvorlage

Wenn Sie eine Startvorlage nicht mehr benötigen, können Sie sie mit dem folgenden [delete-launch-template](#) Befehl löschen. Beim Löschen einer Startvorlage werden alle ihre Versionen gelöscht.

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b729example
```

Aktualisieren einer Auto-Scaling-Gruppe zum Verwenden einer Startvorlage

Sie können den [update-auto-scaling-group](#) Befehl verwenden, um einer vorhandenen Auto Scaling Scaling-Gruppe eine Startvorlage hinzuzufügen.

Aktualisieren einer Auto-Scaling-Gruppe, um die neueste Version einer Startvorlage zu verwenden

Mit dem folgenden [update-auto-scaling-group](#) Befehl wird die angegebene Auto Scaling Scaling-Gruppe aktualisiert, sodass sie die neueste Version der angegebenen Startvorlage verwendet.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--launch-template-id lt-068f72b729example
```

```
--launch-template LaunchTemplateId=lt-068f72b729example,Version='$Latest'
```

Eine Auto-Scaling-Gruppe aktualisieren, um eine bestimmte Version einer Startvorlage zu verwenden

Mit dem folgenden [update-auto-scaling-group](#) Befehl wird die angegebene Auto Scaling Scaling-Gruppe aktualisiert, sodass sie eine bestimmte Version der angegebenen Startvorlage verwendet.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

Verwenden Sie AWS Systems Manager Parameter anstelle von AMI IDs in Startvorlagen

In diesem Abschnitt erfahren Sie, wie Sie eine Startvorlage erstellen, die einen AWS Systems Manager Parameter angibt, der auf eine Amazon Machine Image (AMI) -ID verweist. Sie können einen Parameter verwenden, der in Ihrem eigenen gespeichert ist AWS-Konto, einen Parameter, der von einem anderen gemeinsam genutzt wird AWS-Konto, oder einen öffentlichen Parameter für ein öffentliches AMI, das von verwaltet wird, verwenden AWS.

Mit Systems Manager Manager-Parametern können Sie Ihre Auto Scaling Scaling-Gruppen so aktualisieren, dass sie neues AMI verwenden, IDs ohne bei jeder Änderung einer AMI-ID neue Startvorlagen oder neue Versionen von Startvorlagen erstellen zu müssen. Diese IDs können sich regelmäßig ändern, z. B. wenn ein AMI mit den neuesten Betriebssystem- oder Softwareupdates aktualisiert wird.

Sie können Ihre eigenen Systems Manager Manager-Parameter mithilfe des [Parameterspeichers](#), [einer Funktion von](#), [erstellen](#), aktualisieren oder löschen AWS Systems Manager. Sie müssen einen Systems-Manager-Parameter erstellen, bevor Sie ihn in einer Startvorlage verwenden können. Erstellen Sie zunächst einen Parameter mit dem Datentyp `aws:ec2:image` und geben Sie als Wert die ID eines AMI ein. Die AMI-ID nimmt Form `ami-<identifizier>` an, zum Beispiel `ami-123example456`. Die korrekte AMI-ID ist vom Instance-Typ und der AWS-Region abhängig, in der Sie Ihre Auto-Scaling-Gruppe starten.

Weitere Informationen zum Erstellen eines gültigen Parameters für eine AMI-ID finden Sie unter [Systems Manager Manager-Parameter erstellen](#).

Erstellen Sie eine Startvorlage, die einen Parameter für das AMI angibt

Verwenden Sie eine der folgenden Methoden, um eine Startvorlage zu erstellen, die einen Parameter für das AMI angibt:

Console

Um eine Startvorlage mithilfe eines AWS Systems Manager Parameters zu erstellen

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich Launch Templates (Startvorlagen) und dann Create launch template (Startvorlage erstellen) aus.
3. Geben Sie für Launch template name (Startvorlagenname) einen aussagekräftigen Namen für die Startvorlage ein.
4. Wählen Sie unter Anwendungs- und Betriebssystem-Images (Amazon Machine Image) die Option Mehr durchsuchen aus AMIs.
5. Klicken Sie auf die Pfeilschaltfläche rechts neben der Suchleiste und wählen Sie dann Benutzerdefinierten value/Systems Manager-Parameter angeben.
6. Gehen Sie im Dialogfeld Benutzerdefinierten Wert oder Systems-Manager-Parameter angeben wie folgt vor:
 - a. Geben Sie für AMI-ID oder Systems-Manager-Parameterzeichenfolge den Systems-Manager-Parameternamen in einem der folgenden Formate ein:

So verweisen Sie auf einen öffentlichen Parameter:

- **resolve:ssm:*public-parameter***

So verweisen Sie auf einen Parameter, der im selben Konto gespeichert ist:

- **resolve:ssm:*parameter-name***
- **resolve:ssm:*parameter-name:version-number***
- **resolve:ssm:*parameter-name:label***

So verweisen Sie auf einen von einem anderen AWS-Konto freigegebenen Parameter:

- **resolve:ssm:*parameter-ARN***

- **resolve:ssm:*parameter-ARN*:*version-number***
- **resolve:ssm:*parameter-ARN*:*Label***

b. Wählen Sie Speichern.

7. Konfigurieren Sie nach Bedarf weitere Startvorlageneinstellungen und wählen Sie dann Startvorlage erstellen aus. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).

AWS CLI

Um eine Startvorlage zu erstellen, die einen Systems Manager Manager-Parameter angibt, können Sie einen der folgenden Beispielbefehle verwenden. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

Beispiel: Erstellen Sie eine Startvorlage, die einen öffentlichen Parameter „AWS-own“ angibt

Verwenden Sie die folgende Syntax: `resolve:ssm:public-parameter`, wobei `resolve:ssm` das Standardpräfix und `public-parameter` der Pfad und Name des öffentlichen Parameters ist.

In diesem Beispiel verwendet die Startvorlage einen von AWS-bereitgestellten öffentlichen Parameter, um Instances mit dem neuesten Amazon Linux 2-AMI in dem zu starten AWS-Region , das für Ihr Profil konfiguriert ist.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
--version-description version1 \
--launch-template-data file://config.json
```

Inhalt von `config.json`:

```
{
  "ImageId": "resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2",
  "InstanceType": "t2.micro"
}
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-089c023a30example",
```

```

    "LaunchTemplateName": "my-template-for-auto-scaling",
    "CreateTime": "2022-12-28T19:52:27.000Z",
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "DefaultVersionNumber": 1,
    "LatestVersionNumber": 1
  }
}

```

Beispiel: Erstellen Sie eine Startvorlage, die einen Parameter spezifiziert, der im selben Konto gespeichert ist

Verwenden Sie die folgende Syntax: `resolve:ssm:parameter-name`, wobei `resolve:ssm` das Standardpräfix und `parameter-name` der Systems-Manager-Parametername ist.

Im folgenden Beispiel wird eine Startvorlage erstellt, die die AMI-ID aus einem vorhandenen Systems-Manager-Parameter mit dem Namen `golden-ami` abrufen.

```

aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling \
  --launch-template-data file://config.json

```

Inhalt von `config.json`:

```

{
  "ImageId": "resolve:ssm:golden-ami",
  "InstanceType": "t2.micro"
}

```

Die Standardversion des Parameters ist, falls nicht angegeben, die neueste Version.

Das folgende Beispiel verweist auf eine bestimmte Version des `golden-ami`-Parameters. Das Beispiel verwendet Version `3` des `golden-ami`-Parameters, Sie können jedoch jede gültige Versionsnummer verwenden.

```

{
  "ImageId": "resolve:ssm:golden-ami:3",
  "InstanceType": "t2.micro"
}

```

Das folgende ähnliche Beispiel verweist auf die Parameterbezeichnung `prod`, die einer bestimmten Version des `golden-ami`-Parameters zugeordnet ist.

```
{
  "ImageId":"resolve:ssm:golden-ami:prod",
  "InstanceType":"t2.micro"
}
```

Es folgt eine Beispielausgabe.

```
{
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-068f72b724example",
    "LaunchTemplateName": "my-template-for-auto-scaling",
    "CreateTime": "2022-12-27T17:11:21.000Z",
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "DefaultVersionNumber": 1,
    "LatestVersionNumber": 1
  }
}
```

Beispiel: Erstellen Sie eine Startvorlage, die einen Parameter angibt, der von einem anderen gemeinsam genutzt wird AWS-Konto

Verwenden Sie die folgende Syntax:`resolve:ssm:parameter-ARN`, wobei `resolve:ssm` das Standardpräfix und `parameter-ARN` der ARN des Systems Manager Manager-Parameters sind.

Im folgenden Beispiel wird eine Startvorlage erstellt, die die AMI-ID aus einem vorhandenen Systems Manager Manager-Parameter mit dem ARN von abrufen `arn:aws:ssm:us-east-2:123456789012:parameter/MyParameter`.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
--version-description version1 \
--launch-template-data file://config.json
```

Inhalt von `config.json`:

```
{
  "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/MyParameter",
  "InstanceType":"t2.micro"
}
```

Die Standardversion des Parameters ist, falls nicht angegeben, die neueste Version.

Das folgende Beispiel verweist auf eine bestimmte Version des *MyParameter*-Parameters. Das Beispiel verwendet Version *3* des *MyParameter*-Parameters, Sie können jedoch jede gültige Versionsnummer verwenden.

```
{
  "ImageId": "resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
  MyParameter:3",
  "InstanceType": "t2.micro"
}
```

Das folgende ähnliche Beispiel verweist auf die Parameterbezeichnung *prod*, die einer bestimmten Version des *MyParameter*-Parameters zugeordnet ist.

```
{
  "ImageId": "resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
  MyParameter:prod",
  "InstanceType": "t2.micro"
}
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-00f93d4588example",
    "LaunchTemplateName": "my-template-for-auto-scaling",
    "CreateTime": "2024-01-08T12:43:21.000Z",
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "DefaultVersionNumber": 1,
    "LatestVersionNumber": 1
  }
}
```

Um einen Parameter aus dem Parameterspeicher in einer Startvorlage anzugeben, benötigen Sie die `ssm:GetParameters` Berechtigung für den angegebenen Parameter. Jeder, der die Startvorlage verwendet, benötigt auch die `ssm:GetParameters` Erlaubnis, damit der Parameterwert validiert werden kann. Weitere Informationen finden Sie unter [Beschränken des Zugriffs auf Systems Manager Manager-Parameter mithilfe von IAM-Richtlinien](#) im AWS Systems Manager Benutzerhandbuch.

Stellen Sie sicher, dass eine Startvorlage die richtige AMI-ID erhält

Verwenden Sie den [describe-launch-template-versions](#) Befehl und fügen Sie die `--resolve-alias` Option hinzu, um den Parameter in die tatsächliche AMI-ID aufzulösen.

```
aws ec2 describe-launch-template-versions --launch-template-name my-template-for-auto-scaling \  
--versions 1 --resolve-alias
```

Das Beispiel gibt die AMI-ID für ImageId zurück. Wenn eine Instance mit dieser Startvorlage gestartet wird, wird die AMI-ID zu `ami-0ac394d6a3example` aufgelöst.

```
{  
  "LaunchTemplateVersions": [  
    {  
      "LaunchTemplateId": "lt-089c023a30example",  
      "LaunchTemplateName": "my-template-for-auto-scaling",  
      "VersionNumber": 1,  
      "CreateTime": "2022-12-28T19:52:27.000Z",  
      "CreatedBy": "arn:aws:iam::123456789012:user/Bob",  
      "DefaultVersion": true,  
      "LaunchTemplateData": {  
        "ImageId": "ami-0ac394d6a3example",  
        "InstanceType": "t2.micro",  
      }  
    }  
  ]  
}
```

Zugehörige Ressourcen

Weitere Informationen zur Angabe eines Systems Manager Manager-Parameters in Ihrer Startvorlage finden Sie unter [Verwenden eines Systems Manager Manager-Parameters anstelle einer AMI-ID](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zum Arbeiten mit Systems-Manager-Parametern finden Sie in den folgenden Referenzmaterialien in der Systems-Manager-Dokumentation.

- Informationen zum Erstellen von Parameterversionen und Labels finden Sie unter [Arbeiten mit Parameterversionen](#) und [Arbeiten mit Parameterbeschriftungen](#).

- Informationen zum Nachschlagen der öffentlichen AMI-Parameter, die von Amazon EC2 unterstützt werden, finden Sie unter [Öffentliche AMI-Parameter aufrufen](#).
- Informationen zur gemeinsamen Nutzung von Parametern mit anderen AWS Konten oder über AWS Organizations finden Sie unter [Arbeiten mit gemeinsam genutzten Parametern](#).
- Informationen zur Überwachung, ob Ihre Parameter erfolgreich erstellt wurden, finden Sie unter [Native Parameterunterstützung für Amazon Machine Image IDs](#).

Einschränkungen

Beachten Sie bei der Arbeit mit Systems Manager Manager-Parametern die folgenden Einschränkungen:

- Amazon EC2 Auto Scaling unterstützt nur die Angabe von AMI IDs als Parameter.
- Das Erstellen oder Aktualisieren [gemischter Instanzgruppen](#) mit [attributbasierter Instanztypauswahl](#) mithilfe einer Startvorlage, die einen Systems Manager Manager-Parameter angibt, wird nicht unterstützt.
- Wenn Ihre Auto Scaling Scaling-Gruppe eine Startvorlage verwendet, die einen Systems Manager Manager-Parameter spezifiziert, können Sie eine Instance-Aktualisierung nicht mit der gewünschten Konfiguration oder mithilfe von Skip Matching starten.
- Wenn Ihre Auto Scaling Scaling-Gruppe eine Startvorlage verwendet, die einen Systems Manager Manager-Parameter angibt, werden Warm-Pools nicht unterstützt.
- Bei jedem Aufruf zur Erstellung oder Aktualisierung Ihrer Auto-Scaling-Gruppe löst Amazon EC2 Auto Scaling den Systems-Manager-Parameter in der Startvorlage auf. Wenn Sie erweiterte Parameter oder höhere Durchsatzgrenzen verwenden, können die häufigen Aufrufe des Parameterspeichers (d. h. der `GetParameters`-Vorgang) Ihre Kosten für Systems Manager erhöhen, da Gebühren pro Parameterspeicher-API-Interaktion anfallen. Weitere Informationen finden Sie unter [AWS Systems Manager Preise](#).

Auto Scaling Scaling-Startkonfigurationen

Important

Einschränkungen:

- Ab dem 1. Januar 2023 werden neue Amazon EC2 EC2-Instance-Typen in Startkonfigurationen nicht mehr unterstützt. Dies beinhaltet die Unterstützung für alle Instance-Typen, die AWS-Region nach dem Start der ersten Region hinzugefügt wurden.
- Konten, die am oder nach dem 1. Juni 2023 erstellt wurden, können keine neuen Startkonfigurationen über die Konsole erstellen.
- Konten, die am oder nach dem 1. Oktober 2024 erstellt wurden, können mit keiner Methode (Konsole AWS CLI, API oder CloudFormation) neue Startkonfigurationen erstellen.

Migrieren Sie zu Startvorlagen, um sicherzustellen, dass Sie weder jetzt noch in future neue Startkonfigurationen erstellen müssen. Informationen zum Migrieren Ihrer Auto-Scaling-Gruppen zu Startvorlagen finden Sie unter [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).

Note

Möglicherweise können Sie eine Startkonfiguration mit einem Instance-Typ erstellen, der in einer Region nicht mehr unterstützt wird. Wir empfehlen Ihnen, zu Startvorlagen zu migrieren.

Wir stellen Informationen zu Startkonfigurationen für Kunden bereit, die noch nicht von Startkonfigurationen zu Startvorlagen migriert sind. Eine Startkonfiguration ist eine Instance-Konfigurationsvorlage, die eine Auto-Scaling-Gruppe zum Starten von EC2-Instances verwendet. Beim Erstellen einer Startkonfiguration geben Sie Informationen für die Instances an. Fügen Sie die ID des Amazon Machine Image (AMI), den Instance-Typ, ein Schlüsselpaar, mindestens eine Sicherheitsgruppe und eine Blockgerät-Zuweisung hinzu. Beim Start einer EC2-Instance geben Sie die gleichen Informationen an.

Sie können die Startkonfiguration für mehrere Auto-Scaling-Gruppen verwenden. Sie können jedoch nur eine Startkonfiguration pro Auto-Scaling-Gruppe angeben und diese nach der Erstellung nicht

mehr ändern. Um die Startkonfiguration für eine Auto-Scaling-Gruppe zu ändern, müssen Sie eine Startkonfiguration erstellen und dann damit Ihre Auto-Scaling-Gruppe aktualisieren.

Inhalt

- [Erstellen einer Startkonfiguration](#)
- [Ändern der Startkonfiguration für eine Auto-Scaling-Gruppe](#)

Erstellen einer Startkonfiguration

Important

Einschränkungen:

- Ab dem 1. Januar 2023 werden neue Amazon EC2 EC2-Instance-Typen in Startkonfigurationen nicht mehr unterstützt. Dies beinhaltet die Unterstützung für alle Instance-Typen, die AWS-Region nach dem Start der ersten Region hinzugefügt wurden.
- Konten, die am oder nach dem 1. Juni 2023 erstellt wurden, können keine neuen Startkonfigurationen über die Konsole erstellen.
- Konten, die am oder nach dem 1. Oktober 2024 erstellt wurden, können mit keiner Methode (Konsole AWS CLI, API oder CloudFormation) neue Startkonfigurationen erstellen.

Migrieren Sie zu Startvorlagen, um sicherzustellen, dass Sie weder jetzt noch in future neue Startkonfigurationen erstellen müssen. Informationen zum Migrieren Ihrer Auto-Scaling-Gruppen zu Startvorlagen finden Sie unter [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).

Note

Möglicherweise können Sie eine Startkonfiguration mit einem Instance-Typ erstellen, der in einer Region nicht mehr unterstützt wird. Wir empfehlen Ihnen, zu Startvorlagen zu migrieren.

In diesem Thema wird beschrieben, wie Sie eine Startkonfiguration erstellen. Wir stellen Informationen zu Startkonfigurationen für Kunden bereit, die noch nicht von Startkonfigurationen zu Startvorlagen migriert sind.

Nachdem Sie eine Startkonfiguration erstellt haben, können Sie sie nicht mehr ändern. Stattdessen müssen Sie eine neue Startkonfiguration erstellen.

Informationen zum Zuordnen einer neuen Startkonfiguration zu einer vorhandenen Auto Scaling Scaling-Gruppe finden Sie unter [Ändern der Startkonfiguration für eine Auto-Scaling-Gruppe](#).

Informationen zum Erstellen einer neuen Auto Scaling Scaling-Gruppe finden Sie unter [Eine Auto-Scaling-Gruppe mithilfe einer Startkonfiguration erstellen](#).

Inhalt

- [Erstellen einer Startkonfiguration](#)
- [Konfigurieren der Instance-Metadaten-Optionen](#)
- [Erstellen einer Startkonfiguration aus einer EC2-Instance](#)

Erstellen einer Startkonfiguration


So erstellen Sie eine Startkonfiguration (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie in der oberen Navigationsleiste Ihre AWS Region aus.
3. Wählen Sie im linken Navigationsbereich unter Auto Scaling Auto-Scaling-Gruppen aus.
4. Wählen Sie oben auf der Seite Startkonfigurationen aus. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Startkonfigurationen anzeigen aus, um zu bestätigen, dass Sie die Seite Startkonfigurationen aufrufen möchten.
5. Klicken Sie auf Erstellen einer Startkonfiguration und geben Sie einen Namen für die Startkonfiguration ein.
6. Wählen Sie für Amazon Machine Image (AMI) ein AMI aus. Um ein bestimmtes AMI zu finden, können Sie [ein passendes AMI finden](#), sich die ID notieren und als Suchkriterium eingeben.

So rufen Sie die ID des Amazon Linux 2-AMI ab:

- a. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
 - b. Wählen Sie im Navigationsbereich unter Instances Instances und dann Instances starten aus.
 - c. Notieren Sie auf der Registerkarte Schnellstart der Seite Auswählen eines Amazon Machine Image die ID des AMI neben Amazon Linux 2-AMI.
7. Für Instance-Typ wählen Sie eine Hardware-Konfiguration für Ihre Instances aus.

8. Unter Zusätzliche Konfiguration achten Sie auf die folgenden Felder:
 - a. (Optional) Für Kaufoption können Sie Spot-Instances anfordern auswählen, um Spot-Instances zum aktuellen Spot-Preis anzufordern, dessen Obergrenze der On-Demand-Preis ist. Optional können Sie einen Höchstpreis pro Spot-Instance-Stunde angeben.

 Note

Spot-Instances sind eine kostengünstige Wahl im Vergleich zu On-Demand-Instances, sofern Sie bei der Nutzung Ihrer Anwendungen zeitlich flexibel sind und Unterbrechungen verschmerzen können. Weitere Informationen finden Sie unter [Spot-Instances für fehlertolerante und flexible Anwendungen anfordern](#).


- b. (Optional) Wählen Sie unter IAM-Instance-Profil eine Rolle aus, die mit den Instances verknüpft werden soll. Weitere Informationen finden Sie unter [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#).
 - c. (Optional) Wählen Sie für Monitoring aus, ob die Instances Metrikdaten in Intervallen von 1 Minute an Amazon veröffentlichen können, CloudWatch indem Sie die detaillierte Überwachung aktivieren. Es fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter [Überwachung für Auto-Scaling-Instances konfigurieren](#).
 - d. (Optional) Für Erweiterte Details, Benutzerdaten können Sie Benutzerdaten angeben, um eine Instance während des Starts zu konfigurieren, oder um nach dem Starten der Instance ein Konfigurationsskript auszuführen.
 - e. (Optional) Für Erweiterte Details, IP-Adresstyp, wählen Sie aus, ob Sie den Instances der Gruppe eine [Öffentliche IP-Adresse](#) zuweisen. Wenn Sie keinen Wert festlegen, verwenden Sie standardmäßig die automatische Zuweisung öffentlicher IP-Einstellungen der Subnetze, in denen Ihre Instances gestartet werden.
9. (Optional) Bei Speicher (Volumes) können Sie, wenn Sie keinen zusätzlichen Speicher benötigen, diesen Abschnitt überspringen. Wenn Sie Volumes angeben, die den Instances zusätzlich zu den von AMI angegebenen Volumes hinzugefügt werden sollen, wählen Sie Hinzufügen eines neuen Volumes aus. Wählen Sie dann die gewünschten Optionen und zugeordneten Werte für Geräte, Snapshot, Größe, Volume-Typ, IOPS, Durchsatz, Beim Beenden löschen und Verschlüsselt aus.
10. Für Sicherheitsgruppen erstellen Sie die Sicherheitsgruppe, die den Instances der Gruppe zugeordnet werden soll, oder wählen Sie sie aus. Wenn Sie die Option Erstellen einer neuen Sicherheitsgruppe ausgewählt lassen, wird eine Standard-SSH-Regel für Amazon-EC2-

Instances konfiguriert, auf denen Linux ausgeführt wird. Für Amazon-EC2-Instances, die Windows ausführen, wird eine Standard-RDP-Rolle konfiguriert.

11. Für Schlüsselpaar (Login) wählen Sie eine Option unter Optionen für Schlüsselpaar aus.

Wenn Sie ein Amazon EC2-Instance-Schlüsselpaar bereits konfiguriert haben, können Sie das Schlüsselpaar hier auswählen.

Wenn Sie noch kein Amazon EC2-Instance-Schlüsselpaar haben, klicken Sie auf Create a new key pair (Ein neues Schlüsselpaar erstellen) und geben Sie einen wiedererkennbaren Namen ein. Wählen Sie Download Key Pair (Schlüsselpaar herunterladen) aus, um das Schlüsselpaar auf Ihrem Computer herunterzuladen.

 **Important**

Wählen Sie nicht Proceed without a key pair (Ohne Schlüsselpaar fortfahren) aus, wenn Sie eine Verbindung mit Ihrer Instance herstellen müssen.

12. Aktivieren Sie das Bestätigungskontrollkästchen und wählen Sie dann Create launch configuration (Startkonfiguration erstellen) aus.

Um eine Startkonfiguration aus einer vorhandenen Startkonfiguration (Konsole) zu erstellen

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie in der oberen Navigationsleiste Ihre AWS Region aus.
3. Wählen Sie im linken Navigationsbereich unter Auto Scaling Auto-Scaling-Gruppen aus.
4. Wählen Sie oben auf der Seite Startkonfigurationen aus. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Startkonfigurationen anzeigen aus, um zu bestätigen, dass Sie die Seite Startkonfigurationen aufrufen möchten.
5. Wählen Sie die Startkonfiguration und anschließend Actions, Copy launch configuration aus. So wird eine mit dem Original identische Startkonfiguration (mit Ausnahme des Zusatzes "Kopie" im Namen) erstellt.
6. Bearbeiten Sie auf der Seite Copy Launch Configuration nach Bedarf die Konfigurationsoptionen und wählen Sie anschließend Create launch configuration aus.

Mithilfe der Befehlszeile erstellen Sie eine Startkonfiguration wie folgt:

Verwenden Sie einen der folgenden Befehle:

- [create-launch-configuration](#) (AWS CLI)
- [Neu — ASLaunch Konfiguration](#) (AWS Tools for Windows PowerShell)

Konfigurieren der Instance-Metadaten-Optionen

Amazon EC2 Auto Scaling unterstützt die Konfiguration des Instance Metadata Service (IMDS) in Startkonfigurationen. Dadurch haben Sie die Möglichkeit, Startkonfigurationen zu verwenden, um die Amazon EC2 EC2-Instances in Ihren Auto Scaling Scaling-Gruppen so zu konfigurieren, dass sie Instance Metadata Service Version 2 (IMDSv2) benötigen, eine sitzungsorientierte Methode zum Anfordern von Instance-Metadaten. Einzelheiten zu IMDSv2 den Vorteilen finden Sie in diesem AWS Blog-Artikel über [Verbesserungen, mit denen der EC2-Instance-Metadaten-Service noch besser geschützt](#) werden kann.

Sie können IMDS so konfigurieren, dass es IMDSv2 sowohl als auch IMDSv1 (Standard) unterstützt oder die Verwendung von vorschreibt. IMDSv2 Wenn Sie das AWS CLI oder eines der beiden verwenden, SDKs um IMDS zu konfigurieren, müssen Sie die neueste Version des AWS CLI oder des SDK verwenden, um die Verwendung von zu verlangen. IMDSv2

Sie können Ihre Startkonfiguration wie folgt konfigurieren:

- Erfordern die Verwendung von, IMDSv2 wenn Instanz-Metadaten angefordert werden
- Angeben des PUT-Antwort-Hop-Limits
- Deaktivieren des Zugriffs auf Instance-Metadaten

Weitere Informationen zur Konfiguration des Instance-Metadaten-Service finden Sie im folgenden Thema: [Konfiguration des Instance-Metadaten-Service](#) im Amazon EC2 EC2-Benutzerhandbuch.

Konfigurieren Sie mit den folgenden Schritten IMDS-Optionen in einer Startkonfiguration. Nach dem Erstellen Ihrer Startkonfiguration können Sie diese mit Ihrer Auto-Scaling-Gruppe verknüpfen. Wenn Sie die Startkonfiguration einer vorhandenen Auto-Scaling-Gruppe zuordnen, wird die vorhandene Startkonfiguration von der Auto-Scaling-Gruppe getrennt, und vorhandene Instances müssen ersetzt werden, um die IMDS-Optionen zu verwenden, die Sie in der neuen Startkonfiguration angegeben haben. Weitere Informationen finden Sie unter [Ändern der Startkonfiguration für eine Auto-Scaling-Gruppe](#).

So konfigurieren Sie IMDS in einer Startkonfiguration (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.

2. Wählen Sie in der oberen Navigationsleiste Ihre AWS Region aus.
3. Wählen Sie im linken Navigationsbereich unter Auto Scaling Auto-Scaling-Gruppen aus.
4. Wählen Sie oben auf der Seite Startkonfigurationen aus. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Startkonfigurationen anzeigen aus, um zu bestätigen, dass Sie die Seite Startkonfigurationen aufrufen möchten.
5. Klicken Sie auf Erstellen einer Startkonfiguration und erstellen Sie die Startkonfiguration wie gewohnt. Enthalten sind die ID des Amazon Machine Image (AMI), der Instance-Typ und optional ein Schlüsselpaar, mindestens eine Sicherheitsgruppe und alle zusätzlichen EBS-Volumes oder Instance-Speicher-Volumes für Ihre Instances.
6. Um Instance-Metadatenoptionen für alle Instances zu konfigurieren, die dieser Startkonfiguration zugeordnet sind, finden Sie bei Zusätzliche Konfiguration unter Erweiterte Details wie folgt:
 - a. Für Metadata accessible (Metadaten zugänglich) wählen Sie aus, ob der Zugriff auf den HTTP-Endpunkt des Instance-Metadatenservices aktiviert oder deaktiviert werden soll. Standardmäßig ist der HTTP-Endpunkt aktiviert. Wenn Sie den Endpunkt deaktivieren, wird der Zugriff auf Ihre Instance-Metadaten deaktiviert. Sie können die Bedingung so angeben, dass sie IMDSv2 nur erforderlich ist, wenn der HTTP-Endpunkt aktiviert ist.
 - b. Für die Metadatenversion können Sie festlegen, dass bei der Anforderung von Instanz-Metadaten die Verwendung von Instance Metadata Service Version 2 (IMDSv2) vorgeschrieben ist. Wenn Sie keinen Wert angeben, werden standardmäßig IMDSv1 sowohl als auch unterstützt IMDSv2.
 - c. Für Metadata token response hop limit (Antwort-Hop-Limit des Metadaten-Tokens) können Sie die zulässige Anzahl von Netzwerk-Hops für das Metadaten-Token festlegen. Wenn Sie keinen Wert angeben, wird der Standard auf 1 festgelegt.
7. Wählen Sie danach Erstellen einer Startkonfiguration aus.

Um die Verwendung von IMDSv2 in einer Startkonfiguration mit dem zu verlangen AWS CLI

Verwenden Sie den folgenden [create-launch-configuration](#) Befehl mit der `--metadata-options` Einstellung auf `HttpTokens=required`. Wenn Sie einen Wert für `HttpTokens` angeben, müssen Sie auch `HttpEndpoint` aktivieren. Da der Secure Token-Header auf Erforderlich für Anfragen zum Abrufen von Metadaten gesetzt ist, wird in der Instanz entschieden, dass er bei der Anforderung von Instanz-Metadaten verwendet IMDSv2 werden muss.

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc-with-imdsv2 \  
  --metadata-options HttpTokens=required
```

```
--image-id ami-01e24be29428c15b2 \  
--instance-type t2.micro \  
...  
--metadata-options "HttpEndpoint=enabled,HttpTokens=required"
```

So deaktivieren Sie den Zugriff auf Instance-Metadaten

Verwenden Sie den folgenden [create-launch-configuration](#) Befehl, um den Zugriff auf Instanz-Metadaten zu deaktivieren. Sie können den Zugriff später wieder aktivieren, indem Sie den [modify-instance-metadata-options](#) Befehl verwenden.

```
aws autoscaling create-launch-configuration \  
--launch-configuration-name my-lc-with-imsd-disabled \  
--image-id ami-01e24be29428c15b2 \  
--instance-type t2.micro \  
...  
--metadata-options "HttpEndpoint=disabled"
```

Erstellen einer Startkonfiguration aus einer EC2-Instance

Sie haben auch die Möglichkeit, eine Startkonfiguration mit den Attributen einer laufenden EC2-Instance zu erstellen.

Das Erstellen einer Startkonfiguration von Grund auf ist nicht dasselbe wie das Erstellen einer Startkonfiguration aus einer vorhandenen EC2-Instance. Bei der Erstellung einer Startkonfiguration von Grund auf geben Sie Image-ID, Instance-Typ, optionale Ressourcen (z. B. Speichergeräte) und optionale Einstellungen (z. B. Überwachung) an. Beim Erstellen einer Startkonfiguration aus einer ausgeführten Instance übernimmt Amazon EC2 Auto Scaling die Attribute für die Startkonfiguration von der angegebenen Instance. Attribute werden auch aus der Blockgerät-Zuweisung für das AMI abgeleitet, von dem die Instance gestartet wurde; dabei werden alle weiteren Blockgeräte, die nach dem Start hinzugefügt wurden, ignoriert.

Bei der Erstellung einer Startkonfiguration mithilfe einer laufenden Instance können Sie die folgenden Attribute überschreiben, indem Sie sie als Teil derselben Anforderung markieren: AMI, Blockgeräte, Schlüsselpaar, Instance-Profil, Instance-Typ, Kernel, Instance-Überwachung, Platzierungs-Tenancy, Ramdisk, Sicherheitsgruppen, (max) Spot-Preis, Benutzerdaten, ob die Instance über eine öffentliche IP-Adresse verfügt und ob die Instance für EBS optimiert ist.

Note

Weist die angegebene Instance Eigenschaften auf, die derzeit von Startkonfigurationen nicht unterstützt werden, dann sind die von der Auto-Scaling-Gruppe gestarteten Instances möglicherweise nicht identisch mit der ursprünglichen EC2-Instance.

⚠ Important

Das AMI zum Starten der angegebenen Instance muss noch vorhanden sein.

Themen

- [Erstellen einer Startkonfiguration aus einer EC2-Instance \(AWS CLI\)](#)
- [Erstellen einer Startkonfiguration aus einer Instance und Überschreiben der Blockgeräte \(AWS CLI\)](#)
- [Erstellen einer Startkonfiguration und Überschreiben des Instance-Typs \(AWS CLI\)](#)

Erstellen einer Startkonfiguration aus einer EC2-Instance (AWS CLI)

Verwenden Sie den folgenden [create-launch-configuration](#)-Befehl zum Erstellen einer Startkonfiguration aus einer Instance mit den Attributen der Instance. Alle Blockgeräte, die nach dem Start hinzugefügt wurden, werden ignoriert.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance --instance-id i-a8e09d9c
```

Sie können den folgenden [describe-launch-configurations](#)Befehl verwenden, um die Startkonfiguration zu beschreiben und zu überprüfen, ob ihre Attribute mit denen der Instance übereinstimmen.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance
```

Nachfolgend finden Sie eine Beispielantwort.

```
{  
  "LaunchConfigurations": [  

```

```

    {
      "UserData": null,
      "EbsOptimized": false,
      "LaunchConfigurationARN": "arn",
      "InstanceMonitoring": {
        "Enabled": false
      },
      "ImageId": "ami-05355a6c",
      "CreatedTime": "2014-12-29T16:14:50.382Z",
      "BlockDeviceMappings": [],
      "KeyName": "my-key-pair",
      "SecurityGroups": [
        "sg-8422d1eb"
      ],
      "LaunchConfigurationName": "my-lc-from-instance",
      "KernelId": "null",
      "RamdiskId": null,
      "InstanceType": "t1.micro",
      "AssociatePublicIpAddress": true
    }
  ]
}

```

Erstellen einer Startkonfiguration aus einer Instance und Überschreiben der Blockgeräte (AWS CLI)

Standardmäßig verwendet Amazon EC2 Auto Scaling die Attribute der angegebenen EC2-Instance zum Erstellen der Startkonfiguration. Die Blockgeräte kommen jedoch aus dem AMI, das zum Starten der Instance verwendet wird, nicht aus der Instance. Überschreiben Sie die Blockgerät-Zuweisung der Startkonfiguration, um ihr Blockgeräte hinzuzufügen.

Verwenden Sie den folgenden [create-launch-configuration](#) Befehl, um eine Startkonfiguration mithilfe einer EC2-Instance, aber mit einer benutzerdefinierten Blockgerätezuordnung zu erstellen.

```

aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-bdm --instance-id i-a8e09d9c \
  --block-device-mappings "[{"DeviceName":"/dev/sda1","Ebs":{"SnapshotId":"snap-3decf207"}}, {"DeviceName":"/dev/sdf","Ebs":{"SnapshotId":"snap-eed6ac86"} }]"

```

Verwenden Sie den folgenden [describe-launch-configurations](#) Befehl, um die Startkonfiguration zu beschreiben und sicherzustellen, dass sie Ihre benutzerdefinierte Blockgerätezuordnung verwendet.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-bdm
```

Die folgende Beispielantwort beschreibt die Startkonfiguration:

```
{
  "LaunchConfigurations": [
    {
      "UserData": null,
      "EbsOptimized": false,
      "LaunchConfigurationARN": "arn",
      "InstanceMonitoring": {
        "Enabled": false
      },
      "ImageId": "ami-c49c0dac",
      "CreatedTime": "2015-01-07T14:51:26.065Z",
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": {
            "SnapshotId": "snap-3decf207"
          }
        },
        {
          "DeviceName": "/dev/sdf",
          "Ebs": {
            "SnapshotId": "snap-eed6ac86"
          }
        }
      ],
      "KeyName": "my-key-pair",
      "SecurityGroups": [
        "sg-8637d3e3"
      ],
      "LaunchConfigurationName": "my-lc-from-instance-bdm",
      "KernelId": null,
      "RamdiskId": null,
      "InstanceType": "t1.micro",
      "AssociatePublicIpAddress": true
    }
  ]
}
```

Erstellen einer Startkonfiguration und Überschreiben des Instance-Typs (AWS CLI)

Standardmäßig verwendet Amazon EC2 Auto Scaling die Attribute der angegebenen EC2-Instance zum Erstellen der Startkonfiguration. Je nach Ihren Anforderungen möchten Sie vielleicht Attribute aus der Instance überschreiben und die benötigten Werte verwenden. Sie können beispielsweise den Instance-Typ überschreiben.

Verwenden Sie den folgenden [create-launch-configuration](#) Befehl, um eine Startkonfiguration mithilfe einer EC2-Instance zu erstellen, jedoch mit einem anderen Instance-Typ (zum Beispiel `t2.medium`) als der Instance (zum Beispiel `t2.micro`).

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-change-type \  
--instance-id i-a8e09d9c --instance-type t2.medium
```

Verwenden Sie den folgenden [describe-launch-configurations](#) Befehl, um die Startkonfiguration zu beschreiben und zu überprüfen, ob der Instance-Typ überschrieben wurde.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-change-type
```

Die folgende Beispielantwort beschreibt die Startkonfiguration:

```
{  
  "LaunchConfigurations": [  
    {  
      "UserData": null,  
      "EbsOptimized": false,  
      "LaunchConfigurationARN": "arn",  
      "InstanceMonitoring": {  
        "Enabled": false  
      },  
      "ImageId": "ami-05355a6c",  
      "CreatedTime": "2014-12-29T16:14:50.382Z",  
      "BlockDeviceMappings": [],  
      "KeyName": "my-key-pair",  
      "SecurityGroups": [  
        "sg-8422d1eb"  
      ],  
      "LaunchConfigurationName": "my-lc-from-instance-change-type",  
      "KernelId": "null",  
    }  
  ]  
}
```

```
        "RamdiskId": null,  
        "InstanceType": "t2.medium",  
        "AssociatePublicIpAddress": true  
    }  
]  
}
```

Ändern der Startkonfiguration für eine Auto-Scaling-Gruppe

Important

Wir stellen Informationen zu Startkonfigurationen für Kunden bereit, die noch nicht von Startkonfigurationen zu Startvorlagen migriert sind. Informationen zum Migrieren Ihrer Auto-Scaling-Gruppen zu Startvorlagen finden Sie unter [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).

In diesem Thema wird beschrieben, wie Sie Ihrer Auto Scaling Scaling-Gruppe eine andere Startkonfiguration zuordnen.

Nachdem Sie die Startkonfiguration geändert haben, werden alle neuen Instances mit den neuen Konfigurationsoptionen gestartet, bestehende Instances sind davon jedoch nicht betroffen. Weitere Informationen finden Sie unter [Aktualisieren von Auto-Scaling-Instances](#).

So ändern Sie die Startkonfiguration einer Auto-Scaling-Gruppe (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im linken Navigationsbereich unter Auto Scaling Auto-Scaling-Gruppen aus.
3. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Wählen Sie auf der Registerkarte Details die Option Konfiguration starten, Bearbeiten aus.
5. Wählen Sie unter Startkonfiguration die Startkonfiguration aus.
6. Wählen Sie Aktualisieren aus, wenn Sie fertig sind.

So ändern Sie die Startkonfiguration für eine Auto Scaling Scaling-Gruppe über die Befehlszeile

Verwenden Sie einen der folgenden Befehle:

- [update-auto-scaling-group](#) (AWS CLI)
- [Aktualisieren- ASAuto ScalingGroup](#) (AWS Tools for Windows PowerShell)

Auto-Scaling-Gruppen

Note

Wenn Sie mit Auto Scaling-Gruppen noch nicht vertraut sind, führen Sie zunächst die Schritte im Tutorial [Erstellen Sie Ihre erste Auto Scaling Scaling-Gruppe](#) durch und sehen Sie, wie eine Auto Scaling Scaling-Gruppe reagiert, wenn eine Instance in der Gruppe beendet wird.

Eine Auto-Scaling-Gruppe enthält eine Sammlung von EC2-Instances, die zur automatischen Skalierung und Verwaltung als logische Gruppierung behandelt werden. Eine Auto-Scaling-Gruppe ermöglicht Ihnen außerdem die Verwendung von Amazon EC2 Auto Scaling-Funktionen wie Ersetzungen im Zuge von Zustandsprüfungen und Skalierungsrichtlinien. Sowohl die Aufrechterhaltung der Anzahl von Instances in einer Auto-Scaling-Gruppe und die automatische Skalierung sind die wichtigsten Funktionen des Amazon EC2 Auto Scaling-Services.

Die Größe einer Auto-Scaling-Gruppe richtet sich nach der Anzahl der Instances, die Sie als die gewünschte Kapazität einstellen. Sie können seine Größe an den Bedarf anpassen, entweder manuell oder durch automatische Skalierung.

Eine Auto-Scaling-Gruppe beginnt mit dem Start einer ausreichenden Anzahl von EC2-Instances, um die gewünschte Kapazität zu erfüllen. Sie erhält diese Anzahl der Instances aufrecht, indem die Instances der Gruppe regelmäßigen Zustandsprüfungen unterzogen werden. Die Auto-Scaling-Gruppe verwendet weiterhin eine feste Anzahl von Instances, selbst wenn eine Instance fehlerhaft wird. Wird eine Instance fehlerhaft, beendet die Gruppe die fehlerhafte Instance und startet eine andere Instance, um sie zu ersetzen. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Sie können Skalierungsrichtlinien zur dynamischen Erhöhung bzw. Verringerung der Anzahl an Instances in der Gruppe verwenden, um wechselnden Bedingungen entgegenzukommen. Wenn die Skalierungsrichtlinie verwendet wird, passt die Auto-Scaling-Gruppe die gewünschte Kapazität der Gruppe zwischen den von Ihnen angegebenen minimalen und maximalen Kapazitätswerten an und startet bzw. beendet die Instances je nach Bedarf. Sie können auch nach einem Zeitplan skalieren. Weitere Informationen finden Sie unter [Wählen Sie Ihre Skalierungsmethode aus](#).

Beim Erstellen einer Auto-Scaling-Gruppe können Sie entweder On-Demand-Instances, Spot-Instances oder beides starten. Sie können nur dann mehrere Kaufoptionen für Ihre Auto Scaling

Scaling-Gruppe angeben, wenn Sie eine verwenden `launch template`. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

Spot Instances ermöglichen den Zugriff auf ungenutzte EC2-Kapazitäten zu deutlich günstigeren Preisen (im Vergleich zu den On-Demand-Preisen). Weitere Informationen finden Sie unter [Amazon-EC2-Spot-Instances](#). Es gibt wesentliche Unterschiede zwischen Spot-Instances und On-Demand-Instances:

- Der Preis für Spot-Instances variiert je nach Bedarf
- Amazon EC2 kann eine einzelne Spot-Instance beenden, wenn sich die Verfügbarkeit oder der Preis für Spot-Instances ändert.

Wird eine Spot-Instance beendet, versucht die Auto-Scaling-Gruppe, eine Ersatz-Instance zu starten, um die gewünschte Kapazität für die Gruppe aufrechtzuerhalten.

Wenn Instances gestartet werden, wird, wenn Sie mehrere Availability Zones angegeben haben, die gewünschte Kapazität über all diese Availability Zones verteilt. Wenn eine Skalierungsaktion erfolgt, behält Amazon EC2 Auto Scaling automatisch das Gleichgewicht über alle von Ihnen angegebenen Availability Zones bei.

Inhalt

- [Erstellen Sie Auto-Scaling-Gruppen mit Startvorlagen](#)
- [Erstellen Sie Auto-Scaling-Gruppen mit Startkonfigurationen](#)
- [Starten Sie Instances synchron](#)
- [Aktualisieren einer Auto-Scaling-Gruppe](#)
- [Tagging von Auto-Scaling-Gruppen und Instances](#)
- [Wartungsrichtlinien für Instances](#)
- [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#)
- [Reduzieren Sie die Latenz für Anwendungen mit langen Startzeiten mithilfe warmer Pools](#)
- [Auto Scaling Scaling-Gruppenzonenverschiebung](#)
- [Verteilung von Availability Zones für Auto-Scaling-Gruppen](#)
- [Instances von Ihrer Auto Scaling Scaling-Gruppe trennen oder anhängen](#)
- [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#)
- [Löschen der Auto-Scaling-Infrastruktur](#)

Erstellen Sie Auto-Scaling-Gruppen mit Startvorlagen

Wenn Sie eine Startvorlage erstellt haben, können Sie eine Auto-Scaling-Gruppe erstellen, die eine Startvorlage als Konfigurationsvorlage für ihre EC2-Instances verwendet. Die Startvorlage gibt Informationen wie die AMI-ID, den Instance-Typ, das Schlüsselpaar, Sicherheitsgruppen und die Blockgerät-Zuweisung für Ihre Instances an. Weitere Informationen zum Erstellen von Startvorlagen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).

Sie müssen über IAM-Berechtigungen verfügen, um eine Auto-Scaling-Gruppe zu erstellen. Sie müssen auch über ausreichende Berechtigungen verfügen, um die serviceverknüpfte Rolle zu erstellen, die Amazon EC2 Auto Scaling verwendet, um Aktionen in Ihrem Namen durchzuführen, falls sie noch nicht existiert. Beispiele für IAM-Richtlinien, die ein Administrator als Referenz für die Erteilung von Berechtigungen verwenden kann, finden Sie unter [Beispiele für identitätsbasierte Richtlinien](#) und [Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling-Gruppen](#).

Inhalt

- [Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage](#)
- [Erstellen einer Auto-Scaling-Gruppe mithilfe des Amazon EC2-Startassistenten](#)
- [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#)

Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage

Wenn Sie eine Auto-Scaling-Gruppe erstellen, müssen Sie die notwendigen Informationen zur Konfiguration der Amazon EC2-Instances, die Availability Zones und VPC-Subnetze für die Instances, die gewünschte Kapazität sowie die minimalen und maximalen Kapazitätsgrenzen angeben.

Um Amazon EC2-Instances zu konfigurieren, die von Ihrer Auto-Scaling-Gruppe gestartet werden, können Sie eine Startvorlage oder eine Startkonfiguration angeben. Das folgende Verfahren veranschaulicht das Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage.

Voraussetzungen

- Sie müssen eine Startvorlage erstellt haben. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).

Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben auf dem Bildschirm dieselbe aus, AWS-Region die Sie bei der Erstellung der Startvorlage verwendet haben.
3. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
4. Gehen Sie auf der Seite Choose launch template or configuration (Startvorlage oder Konfiguration auswählen) folgendermaßen vor:
 - a. Für Auto-Scaling-Gruppenname geben Sie einen Namen für Ihre Auto-Scaling-Gruppe ein.
 - b. Wählen Sie für Launch template (Startvorlage) eine vorhandene Startvorlage aus.
 - c. Wählen Sie unter Launch template version (Version der Startvorlage) aus, ob die Auto-Scaling-Gruppe beim horizontalen Skalieren nach oben die standardmäßige, die neueste oder eine bestimmte Version der Startvorlage verwenden soll.
 - d. Stellen Sie sicher, dass Ihre Startvorlage alle Optionen unterstützt, die Sie verwenden möchten, und wählen Sie dann Next (Weiter) aus.
5. Wenn Sie auf der Seite Instance-Startoptionen auswählen nicht mehrere Instance-Typen verwenden, können Sie den Abschnitt Anforderungen an den Instance-Typ überspringen, um den EC2-Instance-Typ zu verwenden, der in der Startvorlage angegeben ist.

Um mehrere Instance-Typen zu verwenden, siehe [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

6. Wählen Sie unter Netzwerk für VPC eine VPC. Die Auto-Scaling-Gruppe muss in derselben VPC erstellt werden wie die Sicherheitsgruppe, die Sie in Ihrer Startvorlage angegeben haben.
7. Für Availability Zones und Subnets (Subnetze) wählen Sie ein oder mehrere Subnetze in der angegebenen VPC aus. Verwenden Sie Subnetze in mehreren Availability Zones, um eine hohe Verfügbarkeit zu erzielen. Weitere Informationen finden Sie unter [Überlegungen bei der Auswahl von VPC-Subnetzen](#).
8. Wählen Sie für die Availability Zone-Verteilung eine Vertriebsstrategie aus. Weitere Informationen finden Sie unter [Verteilung von Availability Zones für Auto-Scaling-Gruppen](#).
9. Wenn Sie eine Startvorlage mit einem bestimmten Instance-Typ erstellt haben, können Sie mit dem nächsten Schritt fortfahren, um eine Auto-Scaling-Gruppe zu erstellen, die den Instance-Typ in der Startvorlage verwendet.

Alternativ können Sie auch die Option Startvorlage überschreiben wählen, wenn in Ihrer Startvorlage kein Instance-Typ angegeben ist oder wenn Sie mehrere Instance-Typen für die automatische Skalierung verwenden möchten. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

10. Wählen Sie Next (Weiter) aus, um mit dem nächsten Schritt fortzufahren.

Oder akzeptieren Sie die weiteren Standardwerte, und klicken Sie dann auf Skip to review (Mit Prüfen fortfahren).

11. (Optional) Konfigurieren Sie auf der Seite Mit anderen Diensten integrieren die folgenden Optionen und wählen Sie dann Weiter aus:

- a. Wählen Sie für Load Balancing aus, ob Ihre Auto Scaling Scaling-Gruppe an einen Load Balancer angehängt werden soll. Weitere Informationen finden Sie unter [Elastic Load Balancing](#).
- b. Wählen Sie für die Integrationsoptionen von VPC Lattice aus, ob Sie VPC Lattice verwenden möchten. Weitere Informationen finden Sie unter [Steuern Sie den Verkehrsfluss mit einer Zielgruppe von VPC Lattice](#).
- c. Aktivieren Sie für Amazon Application Recovery Controller (ARC) Zonal Shift das Kontrollkästchen, um Zonal Shift zu aktivieren. Weitere Informationen finden Sie unter [Auto Scaling Scaling-Gruppenzonenverschiebung](#).
 - Wenn Sie Zonal Shift aktivieren, wählen Sie für Verhalten bei der Integritätsprüfung die Option Fehlerhaftes ignorieren oder Fehlerhaftes ersetzen aus. Weitere Informationen finden Sie unter [So funktioniert Zonal Shift für Auto Scaling Scaling-Gruppen](#).
- d. Wählen Sie unter Integritätsprüfungen für Zusätzliche Zustandsprüfungsarten die Option Amazon EBS-Zustandsprüfungen aktivieren aus. Weitere Informationen finden Sie unter [Überwachen Sie Auto Scaling Scaling-Instances mit beeinträchtigten Amazon EBS-Volumes mithilfe von Zustandsprüfungen](#).
- e. Geben Sie unter Frist für Zustandsprüfungen die Zeit in Sekunden ein. Diese Zeitspanne gibt an, wie lange Amazon EC2 Auto Scaling warten muss, bevor der Integritätsstatus einer Instance überprüft wird, nachdem sie den InService Status erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartefrist für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).

12. (Optional) Konfigurieren Sie auf der Seite Gruppengröße und Skalierung konfigurieren die folgenden Optionen und wählen Sie dann Weiter aus:

- a. Geben Sie unter Gruppengröße für Gewünschte Kapazität die anfängliche Anzahl von Instances ein, die gestartet werden sollen.
- b. Wenn Ihr neuer Wert für Gewünschte Kapazität unter Skalierung, Skalierungsgrenzen größer als Die gewünschte Mindestkapazität und Die gewünschte Höchstkapazität ist, wird die gewünschte maximale Kapazität automatisch auf den neuen gewünschten Kapazitätswert erhöht. Sie können die Limits bei Bedarf ändern. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).
- c. Wählen Sie für Automatische Skalierung aus, ob Sie eine Skalierungsrichtlinie für die Zielverfolgung erstellen möchten. Sie können diese Richtlinie auch erstellen, nachdem Sie Ihre Auto-Scaling-Gruppe erstellt haben.

Wenn Sie sich für die Skalierungsrichtlinie für die Zielverfolgung entscheiden, befolgen Sie die Anweisungen unter [Erstellen einer Zielverfolgungs-Skalierungsrichtlinie](#), um die Richtlinie zu erstellen.

- d. Wählen Sie unter Instanzwartungsrichtlinie aus, ob Sie eine Instanzwartungsrichtlinie erstellen möchten. Sie können diese Richtlinie auch erstellen, nachdem Sie Ihre Auto-Scaling-Gruppe erstellt haben. Befolgen Sie zum Erstellen der Richtlinie die Anweisungen unter [Festlegen einer Instance-Wartungsrichtlinie](#).
- e. Wählen Sie unter Zusätzliche Kapazitätseinstellungen, Präferenz Kapazitätsreservierung, aus, ob Sie eine Einstellung für Kapazitätsreservierung verwenden möchten. Weitere Informationen finden Sie unter [Reservieren Sie Kapazität in bestimmten Availability Zones mit Kapazitätsreservierungen](#).
- f. Wählen Sie unter Zusätzliche Einstellungen, Instance Scale-In Protection, aus, ob der Instance Scale-In-Schutz aktiviert werden soll. Weitere Informationen finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).
- g. Wählen Sie für Monitoring aus, ob die Erfassung von Gruppenmetriken aktiviert CloudWatch werden soll. Diese Metriken liefern Messwerte, die Indikatoren für ein potenzielles Problem sein können, wie z.B. die Anzahl der abgebrochenen Instances oder die Anzahl der ausstehenden Instances. Weitere Informationen finden Sie unter [Überwachen Sie CloudWatch Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances](#).
- h. Wählen Sie für das standardmäßige Aufwärmen der Instanz diese Option und wählen Sie die Aufwärmzeit für Ihre Anwendung. Wenn Sie eine Auto Scaling-Gruppe mit einer Skalierungsrichtlinie erstellen, verbessert die Standard-Instance-Aufwärmfunktion die CloudWatch Amazon-Metriken, die für die dynamische Skalierung verwendet werden.

Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

13. (Optional) Konfigurieren Sie auf der Seite Benachrichtigungen hinzufügen die Benachrichtigung und wählen Sie dann Weiter. Weitere Informationen finden Sie unter [Amazon SNS SNS-Benachrichtigungsoptionen für Amazon EC2 Auto Scaling](#).
14. (Optional) Wählen Sie auf der Seite „Tags hinzufügen“ die Option Tag hinzufügen aus, geben Sie für jedes Tag einen Tag-Schlüssel und einen Tag-Wert ein und wählen Sie dann Weiter aus. Weitere Informationen finden Sie unter [Tagging von Auto-Scaling-Gruppen und Instances](#).
15. Wählen Sie auf der Seite Review (Prüfen) Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Erstellen Sie wie folgt eine Auto-Scaling-Gruppe über die Befehlszeile:

Verwenden Sie einen der folgenden Befehle:

- [create-auto-scaling-group](#) (AWS CLI)
- [Neu- ASAuto ScalingGroup](#) (AWS Tools for Windows PowerShell)

Erstellen einer Auto-Scaling-Gruppe mithilfe des Amazon EC2-Startassistenten

Das folgende Verfahren zeigt, wie Sie mit dem Assistenten zum Starten von Instances in der Amazon EC2-Konsole eine Auto-Scaling-Gruppe erstellen. Mit dieser Option wird eine Startvorlage automatisch mit bestimmten Konfigurationsdetails aus dem Assistenten zum Starten von Instances gefüllt.

Note

Der Assistent füllt die Auto-Scaling-Gruppe nicht mit der von Ihnen angegebenen Anzahl von Instances, sondern nur die Startvorlage mit der Amazon Machine Image (AMI)-ID und dem Instance-Typ. Verwenden Sie den Assistenten zum Create Auto Scaling group (Auto-Scaling-Gruppe erstellen), um die Anzahl der zu startenden Instances anzugeben.

Ein AMI enthält die für die Konfiguration einer Instance erforderlichen Informationen. Sie können mehrere Instances aus einem einzigen AMI starten, wenn Sie mehrere Instances mit derselben Konfiguration benötigen. Wir empfehlen die Verwendung eines benutzerdefinierten AMI, auf dem Ihre Anwendung bereits installiert ist, um zu vermeiden, dass Ihre Instances

beendet werden, wenn Sie eine Instance neu starten, die zu einer Auto-Scaling-Gruppe gehört. Um ein benutzerdefiniertes AMI mit Amazon EC2 Auto Scaling zu verwenden, müssen Sie zunächst Ihr AMI aus einer benutzerdefinierten Instance erstellen und dann das AMI verwenden, um eine Startvorlage für Ihre Auto-Scaling-Gruppe zu erstellen.

Voraussetzungen


- Sie müssen dort, AWS-Region wo Sie die Auto Scaling Scaling-Gruppe erstellen möchten, ein benutzerdefiniertes AMI erstellt haben. Weitere Informationen finden Sie unter [Erstellen eines AMI](#) im Benutzerhandbuch für Amazon EC2.

Verwenden Sie ein benutzerdefiniertes AMI als Vorlage

In diesem Abschnitt verwenden Sie den Launch Wizard von Amazon EC2, um eine Startvorlage automatisch mit Ihrem benutzerdefinierten AMI auszufüllen. Wenn Sie die Startvorlage von Grund auf neu einrichten möchten oder weitere Informationen zu den Parametern benötigen, die Sie für Ihre Startvorlage konfigurieren können, lesen Sie [So erstellen Sie eine Startvorlage \(Konsole\)](#).

So verwenden Sie ein benutzerdefiniertes AMI als Vorlage

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. In der Navigationsleiste oben auf dem Bildschirm AWS-Region wird der aktuelle Wert angezeigt. Wählen Sie eine Region aus, in der Sie Ihre Auto-Scaling-Gruppe starten möchten.
3. Wählen Sie im Navigationsbereich Instances aus.
4. Wählen Sie Launch instance (Instance starten) aus und gehen Sie folgendermaßen vor:
 - a. Lassen Sie unter Name and tags (Name und Tags) Name (Name) leer. Der Name ist nicht Teil der Daten, die zum Erstellen einer Startvorlage verwendet werden.
 - b. Wählen Sie unter Anwendungs- und Betriebssystem-Images (Amazon Machine Image) die Option Mehr durchsuchen aus, AMIs um den vollständigen AMI-Katalog zu durchsuchen.
 - c. Wählen Sie My AMIs, suchen Sie das AMI, das Sie erstellt haben, und wählen Sie dann Select aus.
 - d. Wählen Sie unter EC2 Instance Type (EC2-Instance-Typ) einen Instance-Typ aus.

 Note

Wählen Sie denselben Instance-Typ, den Sie bei der Erstellung des AMI verwendet haben, oder einen leistungsfähigeren.

- e. Geben Sie auf der rechten Seite des Bildschirms unter Summary (Übersicht), für Number of instances (Anzahl der Instances) eine beliebige Zahl ein. Die Zahl, die Sie hier eingeben, ist nicht wichtig. Sie geben die Anzahl der Instances an, die gestartet werden sollen, wenn Sie die Auto-Scaling-Gruppe erstellen.

Unter dem Feld Number of instances (Anzahl der Instances) wird eine Meldung angezeigt, die besagt *When launching more than 1 instance, consider EC2 Auto Scaling* (Beim Starten von mehr als einer Instance EC2 Auto Scaling berücksichtigen).

- f. Wählen Sie den Hyperlinktext *consider EC Auto Scaling* (EC2 Auto Scaling berücksichtigen).
- g. Wählen Sie im Bestätigungsdialog *Launch into Auto Scaling Group* (In Auto-Scaling-Gruppe starten) *Continue* (Weiter), um zur Seite *Create launch template* (Startvorlage erstellen) zu gelangen, auf der das AMI und der Instance-Typ, die Sie im Assistenten zum Starten der Instance ausgewählt haben, bereits eingetragen sind.

Nachdem Sie *Continue* (Weiter) gewählt haben, öffnet sich die Seite *Create launch template* (Startvorlage erstellen). Gehen Sie folgendermaßen vor, um die Erstellung einer Startvorlage abzuschließen.

Eine Startvorlage erstellen

1. Geben Sie unter *Launch template name and description* (Name und Beschreibung der Startvorlage) einen Namen und eine Beschreibung für die neue Startvorlage ein.
2. (Optional) Wählen Sie unter *Key pair (login)* (Schlüsselpaar (Login)) für *Key pair name* (Schlüsselpaar-Name) den Namen des zuvor erstellten Schlüsselpaars, das Sie für die Verbindung zu Instances verwenden möchten, z. B. über SSH.
3. (Optional) Wählen Sie unter *Network settings* (Netzwerkeinstellungen) für *Security groups* (Sicherheitsgruppen) eine oder mehrere zuvor erstellte [security groups](#) (Sicherheitsgruppen).
4. (Optional) Aktualisieren Sie unter *Configure storage* (Speicher konfigurieren) die Speicherkonfiguration. Die Standardspeicherkonfiguration wird vom AMI und dem Instance-Typ bestimmt.

5. Wenn Sie mit der Konfiguration der Startvorlage fertig sind, wählen Sie Startvorlage erstellen.
6. Wählen Sie auf der Bestätigungsseite Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Erstellen einer Auto-Scaling-Gruppe

Note

Der Rest dieses Themas beschreibt das grundlegende Verfahren zur Erstellung einer Auto-Scaling-Gruppe. Eine Beschreibung der Parameter, die Sie für Ihre Auto-Scaling-Gruppe konfigurieren können, finden Sie unter [Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage](#).

Nachdem Sie Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) gewählt haben, öffnet sich der Assistent Create Auto Scaling group (Auto-Scaling-Gruppe erstellen). Gehen Sie folgendermaßen vor, um eine Auto-Scaling-Gruppe zu erstellen.

So erstellen Sie eine Auto Scaling-Gruppe


1. Geben Sie auf der Seite Startvorlage oder Konfiguration auswählen einen Namen für die Auto-Scaling-Gruppe ein.
2. Die Startvorlage, die Sie erstellt haben, ist bereits für Sie ausgewählt.

Wählen Sie unter Launch template version (Version der Startvorlage) aus, ob die Auto-Scaling-Gruppe beim horizontalen Skalieren nach oben die standardmäßige, die neueste oder eine bestimmte Version der Startvorlage verwenden soll.

3. Wählen Sie Next (Weiter) aus, um mit dem nächsten Schritt fortzufahren.
4. Wenn Sie auf der Seite Instance-Startoptionen auswählen nicht mehrere Instance-Typen verwenden, können Sie den Abschnitt Anforderungen an den Instance-Typ überspringen, um den EC2-Instance-Typ zu verwenden, der in der Startvorlage angegeben ist.

Um mehrere Instance-Typen zu verwenden, siehe [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

5. Wählen Sie unter Netzwerk für VPC eine VPC. Die Auto-Scaling-Gruppe muss in derselben VPC erstellt werden wie die Sicherheitsgruppe, die Sie in Ihrer Startvorlage angegeben haben.

 Tip

Wenn Sie in Ihrer Startvorlage keine Sicherheitsgruppe angegeben haben, werden Ihre Instances mit einer Standardsicherheitsgruppe aus der von Ihnen angegebenen VPC gestartet. Standardmäßig lässt diese Sicherheitsgruppe eingehenden Datenverkehr von externen Netzwerken nicht zu.

6. Für Availability Zones und Subnets (Subnetze) wählen Sie ein oder mehrere Subnetze in der angegebenen VPC aus.
7. Wählen Sie für die Availability Zone-Verteilung eine Vertriebsstrategie aus. Weitere Informationen finden Sie unter [Verteilung von Availability Zones für Auto-Scaling-Gruppen](#).
8. Wählen Sie zweimal Next (Weiter), um zur Seite Configure group size and scaling policies (Gruppengröße und Skalierungsrichtlinien konfigurieren) zu gelangen.
9. Legen Sie unter Gruppengröße die gewünschte Kapazität fest (anfängliche Anzahl von Instances, die sofort nach Erstellen der Auto-Scaling-Gruppe gestartet werden sollen).
10. Wenn im Abschnitt Skalierung unter Skalierungslimits Ihr neuer Wert für die gewünschte Kapazität größer als die gewünschte Mindestkapazität und die gewünschte Höchstkapazität ist, wird die gewünschte Höchstkapazität automatisch auf den neuen Wert für die gewünschte Kapazität erhöht. Sie können die Limits bei Bedarf ändern. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).
11. Wählen Sie Skip to review (Mit Prüfen fortfahren) aus.
12. Wählen Sie auf der Seite Review (Prüfen) Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Nächste Schritte

Sie können überprüfen, ob die Auto-Scaling-Gruppe korrekt erstellt wurde, indem Sie sich den Aktivitätsverlauf ansehen. Auf der Registerkarte Activity (Aktivität) wird unter Activity history (Aktivitätsverlauf) in der Spalte Status angezeigt, ob Ihre Auto-Scaling-Gruppe-Instances erfolgreich gestartet hat. Wenn die Instances nicht starten oder zwar starten, dann aber sofort abbrechen, lesen Sie die folgenden Themen zu möglichen Ursachen und Lösungen:

- [Fehlersuche bei Amazon EC2 Auto Scaling: Startfehler von EC2-Instance](#)
- [Fehlersuche bei Amazon EC2 Auto Scaling: AMI-Probleme](#)
- [Fehlerbehebung bei fehlerhaften Instances in Amazon EC2 Auto Scaling](#)

Sie können nun ein Load Balancer in derselben Region wie Ihre Auto-Scaling-Gruppe einrichten, falls gewünscht. Weitere Informationen finden Sie unter [Verwenden Sie Elastic Load Balancing, um den eingehenden Anwendungsdatenverkehr in Ihrer Auto Scaling Scaling-Gruppe zu verteilen](#).

Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen

Sie können eine Flotte von On-Demand-Instances und Spot-Instances innerhalb einer einzigen Auto-Scaling-Gruppe starten und automatisch skalieren. Zusätzlich zu den Rabatten für die Nutzung von Spot-Instances können Sie Reserved Instances oder Savings Plans nutzen, um Rabatte auf die regulären On-Demand-Instance-Preise zu erhalten. Diese Faktoren helfen Ihnen, Ihre Kosteneinsparungen für EC2-Instances zu optimieren und die gewünschte Skalierung und Leistung für Ihre Anwendung zu erhalten.

Bei Spot-Instances handelt es sich um Kapazitätsreserven, die im Vergleich zum EC2-On-Demand-Preis stark reduziert erhältlich sind. Spot Instances sind eine kostengünstige Wahl, sofern Sie bei der Ausführung Ihrer Anwendungen zeitlich flexibel sind und Unterbrechungen verschmerzen können. Sie können für verschiedene fehlertolerante und flexible Anwendungen verwendet werden. Beispiele hierfür sind statuslose Webserver, API-Endpunkte, Big-Data- und Analyseanwendungen, containerisierte Workloads, CI/CD Pipelines, Hochleistungs- und Hochdurchsatz-Computing (HPC/ HTC), Rendering-Workloads und andere flexible Workloads.

Weitere Informationen finden Sie unter [Kaufoptionen für Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.

Topics

- [Übersicht über die Einrichtung für die Erstellung einer gemischten Instance-Gruppe](#)
- [Zuweisungsstrategien für mehrere Instance-Typen](#)
- [Gruppe mit gemischten Instanzen mithilfe der attributbasierten Instanztypauswahl erstellen](#)
- [Erstellen Sie eine Gruppe mit gemischten Instances, indem Sie die Instance-Typen manuell auswählen](#)
- [Konfigurieren Sie eine Auto Scaling Scaling-Gruppe für die Verwendung von Instanzgewichten](#)
- [Verwenden von mehreren Startvorlagen](#)

Übersicht über die Einrichtung für die Erstellung einer gemischten Instance-Gruppe

Dieses Thema bietet einen Überblick und bewährte Methoden für die Erstellung einer gemischten Auto Scaling Scaling-Instanzgruppe.

Inhalt

- [-Übersicht](#)
- [Flexibilität bezüglich der Instance-Größe](#)
- [Flexibilität bezüglich der Availability Zone](#)
- [Maximaler Spotpreis](#)
- [Proaktiver Kapazitätsausgleich](#)
- [Skalierungsverhalten](#)
- [Regionale Verfügbarkeit von Instance-Typen](#)
- [Zugehörige Ressourcen](#)
- [Einschränkungen](#)

-Übersicht

Es gibt zwei Möglichkeiten zum Erstellen einer Instances-Gruppe mit gemischten Instances:

- [Attributbasierte Auswahl des Instance-Typs](#) — Definieren Sie Ihre Rechenanforderungen, sodass Ihre Instance-Typen automatisch auf der Grundlage ihrer spezifischen Instance-Attribute ausgewählt werden.
- [Manuelle Auswahl des Instance-Typs](#) — Wählen Sie manuell die Instance-Typen aus, die zu Ihrem Workload passen.

Manual selection

In den folgenden Schritten wird beschrieben, wie Sie eine Instances-Gruppe erstellen, indem Sie die Instances-Gruppe manuell auswählen:

1. Wählen Sie eine Startvorlage, die die Parameter zum Starten einer EC2-Instance enthält. Parameter in Startvorlagen sind optional, aber Amazon EC2 Auto Scaling kann eine Instance nicht starten, wenn die Amilong; (AMI) -ID in der Startvorlage fehlt.
2. Wählen Sie die Option zum Überschreiben der Startvorlage.
3. Wählen Sie manuell die Instance-Typen aus, die zu Ihrem Workload passen.
4. Geben Sie die Prozentsätze der On-Demand-Instances und Spot Instances an, die gestartet werden sollen.

5. Die folgenden Zuweisungsstrategien bestimmen, wie die Amazon EC2 Auto Scaling-Gruppe Ihre gewünschte Kapazität für On-Demand- und Spot-Kapazität von den möglichen Instance-Typen erfüllt.
6. Wählen Sie die Availability Zones und VPC-Subnetze aus, in denen Sie Ihre Instances starten möchten.
7. Geben Sie die Anfangsgröße der Gruppe (die gewünschte Kapazität) sowie die Mindest- und Maximalgröße der Gruppe an.

Überschreibungen sind erforderlich, um den in der Startvorlage deklarierten Instance-Typ zu überschreiben und mehrere Instance-Typen zu verwenden, die in die eigene Ressourcendefinition der Auto-Scaling-Gruppe eingebettet sind. Weitere Informationen zu den verfügbaren Instance-Typen finden Sie unter [Instance-Typen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Sie können auch die folgenden optionalen Parameter für jeden Instance-Typ konfigurieren:

- `LaunchTemplateSpecification`— Sie können einem Instance-Typ nach Bedarf eine andere Startvorlage zuweisen. Diese Option ist zur Zeit in der Konsole nicht verfügbar. Weitere Informationen finden Sie unter [Verwenden von mehreren Startvorlagen](#).
- `WeightedCapacity`— Sie entscheiden, wie viel die Instance im Vergleich zu den übrigen Instances in Ihrer Gruppe auf die gewünschte Kapazität angerechnet wird. Wenn Sie einen `WeightedCapacity`-Wert für einen Instance-Typ angeben, müssen Sie einen `WeightedCapacity`-Wert für alle Instance-Typen angeben. Standardmäßig wird jede Instance als eine Instance auf Ihre gewünschte Kapazität angerechnet. Weitere Informationen finden Sie unter [Konfigurieren Sie eine Auto Scaling Scaling-Gruppe für die Verwendung von Instanzgewichten](#).

Attribute-based selection

Damit Amazon EC2 Auto Scaling Ihre Instance-Typen automatisch auf der Grundlage ihrer spezifischen Instance-Attribute auswählen kann, erstellen Sie mithilfe der folgenden Schritte eine gemischte Instance-Gruppe, indem Sie Ihre Rechenanforderungen angeben:

1. Wählen Sie eine Startvorlage, die die Parameter zum Starten einer EC2-Instance enthält. Parameter in Startvorlagen sind optional, aber Amazon EC2 Auto Scaling kann eine Instance nicht starten, wenn die `Amilong`; (AMI) -ID in der Startvorlage fehlt.
2. Wählen Sie die Option zum Überschreiben der Startvorlage.

3. Geben Sie Instance-Attribute an, die Ihren Rechenanforderungen entsprechen, z. B. v CPUs - und Speicheranforderungen.
4. Geben Sie die Prozentsätze der On-Demand-Instances und Spot Instances an, die gestartet werden sollen.
5. Die folgenden Zuweisungsstrategien bestimmen, wie die Amazon EC2 Auto Scaling-Gruppe Ihre gewünschte Kapazität für On-Demand- und Spot-Kapazität von den möglichen Instance-Typen erfüllt.
6. Wählen Sie die Availability Zones und VPC-Subnetze aus, in denen Sie Ihre Instances starten möchten.
7. Geben Sie die Anfangsgröße der Gruppe (die gewünschte Kapazität) sowie die Mindest- und Maximalgröße der Gruppe an.

Überschreibungen sind erforderlich, um den in der Startvorlage deklarierten Instance-Typ außer Kraft zu setzen und eine Reihe von Instance-Attributen zu verwenden, die Ihre Rechenanforderungen beschreiben. Informationen zu den unterstützten Attributen finden Sie [InstanceRequirements](#) in der Amazon EC2 Auto Scaling API-Referenz. Alternativ können Sie eine Startvorlage verwenden, die bereits die Definition der Instance-Attribute enthält.

Sie können den `LaunchTemplateSpecification`-Parameter auch innerhalb der `Overrides`-Struktur konfigurieren, um einer Reihe von Instance-Anforderungen nach Bedarf eine andere Startvorlage zuzuweisen. Diese Option ist zur Zeit in der Konsole nicht verfügbar. Weitere Informationen finden Sie [LaunchTemplateOverrides](#) in der Amazon EC2 Auto Scaling API-Referenz.

Standardmäßig legen Sie die Anzahl der Instances als die gewünschte Kapazität Ihrer Auto Scaling-Gruppe fest.

Alternativ können Sie den Wert für die gewünschte Kapazität auf die Anzahl von v CPUs oder die Speichergröße festlegen. Verwenden Sie dazu die `DesiredCapacityType`-Eigenschaft im `CreateAutoScalingGroup` API-Vorgang oder das Dropdown-Feld Gewünschter Kapazitätstyp in der AWS-Managementkonsole. Dies ist eine nützliche Alternative zu [Instance-Gewichten](#).

Flexibilität bezüglich der Instance-Größe

Um die Verfügbarkeit zu erhöhen, stellen Sie Ihre Anwendung für mehrere Instance-Typen bereit. Es hat sich bewährt, mehrere Instance-Typen zu verwenden, um die Kapazitätsanforderungen zu

erfüllen. Dadurch kann Amazon EC2 Auto Scaling einen weiteren Instance-Typ starten, wenn in den ausgewählten Availability Zones nicht genügend Instance-Kapazität zur Verfügung steht.

Falls die Instance-Kapazität bei Spot Instances nicht ausreicht, versucht Amazon EC2 Auto Scaling immer wieder, Instances aus anderen Spot-Instance-Pools zu starten. (Die verwendeten Pools hängen von den von Ihnen ausgewählten Instance-Typen und der Zuweisungsstrategie ab.) Amazon EC2 Auto Scaling hilft Ihnen dabei, die Kosteneinsparungen von Spot Instances zu nutzen, indem Sie sie anstelle von On-Demand-Instances starten.

Wir empfehlen, für jeden Workload über mindestens 10 Instance-Typen hinweg flexibel zu sein. Beschränken Sie sich bei der Auswahl Ihrer Instance-Typen nicht auf die beliebtesten neuen Instance-Typen. Die Wahl von Instance-Typen der früheren Generation führt in der Regel zu weniger Spot-Unterbrechungen, da sie von On-Demand-Kunden weniger nachgefragt werden.

Flexibilität bezüglich der Availability Zone

Wir empfehlen dringend, dass Sie Ihre Auto Scaling-Gruppe auf mehrere Availability Zones verteilen. Mit mehreren Availability Zones können Sie Anwendungen entwerfen, die automatisch zwischen den Zonen umschalten, um die Ausfallsicherheit zu erhöhen.

Ein zusätzlicher Vorteil ist, dass Sie im Vergleich zu Gruppen in einer einzelnen Availability Zone auf einen größeren Amazon EC2-Kapazitätspool zugreifen können. Da die Kapazität für jeden Instance-Typ in jeder Availability Zone unabhängig schwankt, können Sie oft mehr Rechenkapazität mit Flexibilität sowohl für den Instance-Typ als auch für die Availability Zone erhalten.

Weitere Informationen zur Verwendung mehrerer Availability Zones finden Sie unter [Beispiel: Aufteilen von Instances in mehrere Availability Zones](#).

Maximaler Spotpreis

Wenn Sie Ihre Auto Scaling Scaling-Gruppe mit dem AWS CLI oder einem SDK erstellen, können Sie den `SpotMaxPrice` Parameter angeben. Der `SpotMaxPrice`-Parameter bestimmt den Höchstpreis, den Sie für eine Spot-Instance-Stunde zu zahlen bereit sind.

Wenn Sie den `WeightedCapacity`-Parameter in Ihren Overrides (oder "`DesiredCapacityType`": "`vcpu`" oder "`DesiredCapacityType`": "`memory-mib`" auf Gruppenebene) angeben, stellt der Höchstpreis den maximalen Einzelpreis dar, nicht den Höchstpreis für eine ganze Instance.

Wir empfehlen ausdrücklich, keinen Höchstpreis anzugeben. Ihre Anwendung läuft möglicherweise nicht, wenn Sie keine Spot-Instances erhalten, z. B. wenn Ihr Höchstpreis zu niedrig ist. Wenn

Sie keinen Höchstpreis angeben, entspricht der Standardhöchstpreis dem On-Demand-Preis. Sie zahlen nur den Spot-Preis für Spot-Instances, die Sie starten. Sie erhalten weiterhin die hohen Rabatte von Spot Instances. Diese Rabatte sind dank der stabilen Spot-Preise des [Spot-Preismodells](#) möglich. Weitere Informationen finden Sie unter [Preise und Einsparungen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Proaktiver Kapazitätsausgleich

Wenn Ihr Anwendungsfall dies zulässt, empfehlen wir Capacity Rebalancing (Kapazitätsausgleich). Capacity Rebalancing hilft Ihnen dabei, die Verfügbarkeit Ihrer Workloads aufrechtzuerhalten, indem Spot-Instances, bei denen das Risiko einer Unterbrechung besteht, proaktiv ersetzt werden.

Wenn Capacity Rebalancing aktiviert ist, versucht Amazon EC2 Auto Scaling, Spot-Instances, die eine Empfehlung zur Neuverteilung von EC2-Instances erhalten haben, proaktiv zu ersetzen. Dies bietet die Möglichkeit, Ihre Arbeitslast auf neue Spot-Instances umzustellen, bei denen kein erhöhtes Unterbrechungsrisiko besteht.

Weitere Informationen finden Sie unter [Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances](#).

Skalierungsverhalten

Wenn Sie eine gemischte Instance-Gruppe erstellen, werden standardmäßig On-Demand-Instances verwendet. Um Spot-Instances verwenden zu können, müssen Sie den Prozentsatz der Gruppe ändern, die als On-Demand-Instances gestartet werden soll. Sie können eine beliebige Zahl zwischen 0 und 100 als On-Demand-Prozentsatz angeben.

Optional können Sie auch eine Basisanzahl von On-Demand-Instances festlegen, mit der begonnen werden soll. Wenn Sie dies tun, wartet Amazon EC2 Auto Scaling mit dem Start von Spot-Instances, bis die Basiskapazität der On-Demand-Instances erreicht ist, sobald die Gruppe aufskaliert. Für alles außerhalb der Basiskapazität werden die On-Demand-Prozentsätze verwendet, um zu bestimmen, wie viele On-Demand-Instances und Spot-Instances gestartet werden sollen.

Amazon EC2 Auto Scaling konvertiert den Prozentsatz in die entsprechende Anzahl von Instances. Wenn das Ergebnis eine Bruchzahl ergibt, wird zugunsten der On-Demand-Instances auf die nächste Ganzzahl aufgerundet.

Die folgende Tabelle veranschaulicht das Verhalten der Auto-Scaling-Gruppe, wenn sie sich vergrößert oder verkleinert.

Beispiel: Skalierungsverhalten

Kaufoptionen Gruppengröße und Anzahl der laufenden Instances bei allen Kaufoptionen

10 20 30 40

Beispiel 1: Basis
von 10, 50/50%
On-Demand/
Spot

On-Demand -Instances (Grundmenge)	10	10	10	10
---	----	----	----	----

On-Demand Instances	0	5	10	15
------------------------	---	---	----	----

Spot Instances	0	5	10	15
----------------	---	---	----	----

Beispiel 2: Basis
von 0, 0/100%
On-Demand/
Spot

On-Demand -Instances (Grundmenge)	0	0	0	0
---	---	---	---	---

On-Demand Instances	0	0	0	0
------------------------	---	---	---	---

Spot Instances	10	20	30	40
----------------	----	----	----	----

Beispiel 3: Basis
von 0, 60/40%
On-Demand/
Spot

Kaufoptionen	Gruppengröße und Anzahl der laufenden Instances bei allen Kaufoptionen			
On-Demand -Instances (Grundmenge)	0	0	0	0
On-Demand Instances	6	12	18	24
Spot Instances	4	8	12	16
Beispiel 4: Basis von 0, 100/0% On-Demand/ Spot				
On-Demand -Instances (Grundmenge)	0	0	0	0
On-Demand Instances	10	20	30	40
Spot Instances	0	0	0	0
Beispiel 5: Basis von 12, 0/100% On-Demand/ Spot				
On-Demand -Instances (Grundmenge)	10	12	12	12
On-Demand Instances	0	0	0	0
Spot Instances	0	8	18	28

Wenn die Gruppengröße zunimmt, versucht Amazon EC2 Auto Scaling, Ihre Kapazität gleichmäßig über die angegebenen Availability Zones zu verteilen. Anschließend startet es Instance-Typen entsprechend der angegebenen Zuweisungsstrategie.

Wenn die Gruppengröße abnimmt, identifiziert Amazon EC2 Auto Scaling zunächst, welcher der beiden Typen (Spot oder On-Demand) beendet werden soll. Anschließend wird versucht, Instances auf ausgewogene Weise über Ihre angegebenen Availability Zones hinweg zu beenden. Außerdem wird die Beendigung von Instances auf eine Weise begünstigt, die Ihren Allokationsstrategien näher kommt. Weitere Informationen zu den Richtlinien zum Beenden finden Sie unter [Kündigungsrichtlinien für Amazon EC2 Auto Scaling konfigurieren](#).

Regionale Verfügbarkeit von Instance-Typen

Die Verfügbarkeit von EC2-Instance-Typen hängt von Ihrer AWS-Region ab. So kann es beispielsweise sein, dass die neueste Generation von Instance-Typen in einer bestimmten Region noch nicht verfügbar ist. Aufgrund der regionalen Unterschiede bei der Instance-Verfügbarkeit können Probleme auftreten, sobald Sie programmatische Anfragen stellen, wenn mehrere Instance-Typen in Ihren Overrides in Ihrer Region nicht verfügbar sind. Die Verwendung mehrerer Instance-Typen, die in Ihrer Region nicht verfügbar sind, kann dazu führen, dass die Anfrage vollständig fehlschlägt. Um das Problem zu lösen, wiederholen Sie die Anfrage mit verschiedenen Instance-Typen und stellen Sie sicher, dass jeder Instance-Typ in der Region verfügbar ist. Verwenden Sie den [describe-instance-type-offerings](#) Befehl, um nach Instance-Typen zu suchen, die je nach Standort angeboten werden. Weitere Informationen [finden Sie unter Suchen nach einem Amazon EC2 EC2-Instance-Typ](#) im Amazon EC2 EC2-Benutzerhandbuch.

Zugehörige Ressourcen

Weitere Best Practices für Spot-Instances finden Sie unter [Bewährte Methoden für EC2 Spot](#) im Amazon EC2 EC2-Benutzerhandbuch.

Einschränkungen

Nachdem Sie einer Auto Scaling Scaling-Gruppe mithilfe einer [Richtlinie für gemischte Instanzen](#) Overrides hinzugefügt haben, können Sie die Overrides mit dem `UpdateAutoScalingGroup` API-Aufruf aktualisieren, aber nicht löschen. Um die Überschreibungen vollständig zu entfernen, müssen Sie zunächst die Auto Scaling Scaling-Gruppe so ändern, dass sie eine Startvorlage oder eine Startkonfiguration anstelle einer Richtlinie für gemischte Instanzen verwendet. Anschließend können Sie erneut eine Richtlinie für gemischte Instanzen ohne Überschreibungen hinzufügen.

Zuweisungsstrategien für mehrere Instance-Typen

Wenn Sie mehrere Instance-Typen verwenden, verwalten Sie, wie Amazon EC2 Auto Scaling Ihre On-Demand- und Spot-Kapazität mithilfe der möglichen Instance-Typen erfüllt. Zu diesem Zweck spezifizieren Sie Zuweisungsstrategien und .

Eine Übersicht der bewährten Methoden für eine Gruppe mit gemischten Instanzen finden Sie unter [Übersicht über die Einrichtung für die Erstellung einer gemischten Instance-Gruppe](#).

Inhalt

- [Spot Instances](#)
- [On-Demand Instances](#)
- [Wie funktionieren die Allokationsstrategien mit Gewichten](#)

Spot Instances

Amazon EC2 Auto Scaling bietet die folgenden Allokationsstrategien, die für Spot-Instances verwendet werden können:

price-capacity-optimized (empfohlen)

Die preis- und kapazitätsoptimierte Zuweisungsstrategie betrachtet sowohl den Preis als auch die Kapazität, um die Spot-Instance-Pools auszuwählen, die am unwahrscheinlichsten unterbrochen werden und den niedrigstmöglichen Preis haben.

Wir empfehlen diese Strategie für den Einstieg. Weitere Informationen finden Sie im AWS Blog unter [Einführung in die price-capacity-optimized Zuweisungsstrategie für EC2-Spot-Instances](#).

capacity-optimized

Amazon EC2 Auto Scaling fordert Ihre Spot Instance aus dem Pool mit optimaler Kapazität für die Anzahl der zu startenden Instances an.

Bei Spot-Instances ändert sich die Preisgestaltung im Laufe der Zeit basierend auf langfristigen Trends bei Angebot und Nachfrage langsam. Die Kapazität schwankt jedoch in Echtzeit. Bei Anwendung der Strategie capacity-optimized wird Spot-Instances automatisch zu den am besten verfügbaren Pools gestartet, indem Echtzeitdaten zur Kapazität analysiert werden und prognostiziert wird, welche Pools am besten verfügbar sind. Dies trägt dazu bei, mögliche Unterbrechungen für Workloads zu minimieren, bei denen Unterbrechungen ggf. aufgrund des

Neustarts von Aufgaben sowie aufgrund von Checkpointing zu höheren Kosten führen. Um bestimmten Instance-Typen eine höhere Chance zu geben, zuerst zu starten, verwenden Sie `capacity-optimized-prioritized`.

`capacity-optimized-prioritized`

Sie legen die Reihenfolge der Instance-Typen für die Überschreibungen der Startvorlagen von der höchsten bis zur niedrigsten Priorität (vom ersten bis zum letzten in der Liste) fest. Amazon EC2 Auto Scaling berücksichtigt die Prioritäten des Instance-Typs nach bestem Wissen und Gewissen, optimiert jedoch zuerst die Kapazität. Dies ist eine gute Option für Workloads, bei denen die Möglichkeit von Unterbrechungen minimiert werden muss, aber auch die Präferenz für bestimmte Instance-Typen von Bedeutung ist. Wenn die On-Demand-Zuweisungsstrategie auf `prioritized` festgelegt ist, wird bei der Abdeckung von On-Demand-Kapazität die gleiche Priorität angewendet.

`lowest-price` (wird nicht empfohlen)

Warning

Wir empfehlen diese `lowest-price` Strategie nicht, da sie das höchste Unterbrechungsrisiko für Ihre Spot-Instances birgt.

Amazon EC2 Auto Scaling fordert Ihre Spot Instances unter Verwendung der preisgünstigsten Pools innerhalb einer Availability Zone für die Anzahl N von Spot-Pools an, die Sie für die Einstellung Pools mit dem niedrigsten Preis angegeben haben. Wenn Sie also beispielsweise vier Instance-Typen und vier Availability Zones angeben, kann Ihre Auto-Scaling-Gruppe auf bis zu 16 Spot-Pools zugreifen. (Vier in jeder Availability Zone.) Wenn Sie für die Zuweisungsstrategie zwei Spot-Pools ($N=2$) angeben, kann Ihre Auto-Scaling-Gruppe die beiden preisgünstigsten Pools pro Availability Zone nutzen, um Ihre Spot-Kapazität abzudecken.

Die `lowest-price` Strategie ist nur verfügbar, wenn Sie die verwenden AWS CLI.

Amazon EC2 Auto Scaling versucht, Spot Instances aus der Anzahl (N) der Pools zu ziehen, die Sie angeben. Wenn einem Pool die Spot-Kapazität ausgeht, bevor Ihre gewünschte Kapazität erreicht ist, wird Amazon EC2 Auto Scaling Ihre Anfrage weiterhin erfüllen, indem sie sie aus dem nächstgünstigsten Pool zieht. Um Ihre gewünschte Kapazität abzudecken, erhalten Sie möglicherweise Spot Instances aus mehr Pools als der von Ihnen angegebenen Anzahl (N). Wenn die meisten Pools keine Spot-Kapazität haben, erhalten Sie Ihre volle gewünschte Kapazität möglicherweise von weniger als der von Ihnen angegebenen Anzahl (N) von Pools.

Note

Wenn Sie eine Spot Instance mit aktiviertem [AMD SEV-SNP](#) starten, wird Ihnen eine zusätzliche stündliche Nutzungsgebühr in Höhe von 10 % des [On-Demand-Stundensatzes](#) des ausgewählten Instance-Typs berechnet. Wenn die Zuweisungsstrategie den Preis als Eingabe verwendet, berücksichtigt Amazon EC2 Auto Scaling diese zusätzliche Gebühr nicht; es wird nur der Spot-Preis verwendet.

On-Demand Instances

Amazon EC2 Auto Scaling bietet die folgenden Zuweisungsstrategien, die für On-Demand-Instances verwendet werden können:

lowest-price

Amazon EC2 Auto Scaling stellt automatisch den günstigsten Instance-Typ in jeder Availability Zone bereit, basierend auf dem aktuellen On-Demand-Preis.

Um Ihre gewünschte Kapazität zu erreichen, erhalten Sie in den einzelnen Availability Zones möglicherweise On-Demand-Instances von mehreren Instance-Typen. Dies hängt davon ab, wie viel Kapazität Sie anfordern.

prioritized

Bei der Abdeckung der On-Demand-Kapazität bestimmt Amazon EC2 Auto Scaling, welcher Instance-Typ zuerst verwendet wird (basierend auf der Reihenfolge der Instance-Typen in der Liste der Startvorlagen-Überschreibungen). Ein Beispiel: Angenommen, Sie geben drei Startvorlagen-Überschreibungen in der folgenden Reihenfolge an: `c5.large`, `c4.large` und `c3.large`. Beim Start Ihrer On-Demand-Instances verwendet die Auto-Scaling-Gruppe erst `c5.large`, dann `c4.large` und schließlich `c3.large`, um die On-Demand-Kapazität abzudecken.

Berücksichtigen Sie beim Verwalten der Prioritätsreihenfolge Ihrer On-Demand-Instances Folgendes:

- Sie können für die Nutzung im Voraus bezahlen, um erhebliche Rabatte für On-Demand-Instances zu erhalten, indem Sie entweder Savings Plans oder Reserved Instances verwenden. Weitere Informationen finden Sie auf der [Preisseite für Amazon EC2](#).
- Bei Reserved Instances gilt die ermäßigte Rate der regulären On-Demand-Instance-Preise, wenn Amazon EC2 Auto Scaling passende Instance-Typen startet. Das bedeutet: Wenn Sie

über ungenutzte Reserved Instances für `c4.large` verfügen, können Sie die Priorität der Instance-Typen so festlegen, dass dem Instance-Typ `c4.large` die höchste Priorität für Ihre Reserved Instances eingeräumt wird. Wenn eine `c4.large`-Instance gestartet wird, erhalten Sie die Preise für die Reserved Instance.

- Bei Savings Plans gelten bei Verwendung von Amazon EC2 Instance Savings Plans oder Compute Savings Plans die regulären On-Demand-Instance-Preise. Savings Plans bieten mehr Flexibilität bei der Priorisierung Ihrer Instance-Typen. Solange Sie Instance-Typen verwenden, die von Ihren Savings Plans abgedeckt werden, können Sie sie in beliebiger Prioritätsreihenfolge festlegen. Sie können auch gelegentlich die gesamte Reihenfolge Ihrer Instance-Typen ändern und trotzdem den ermäßigten Tarif für Savings Plans erhalten. Weitere Informationen zu Savings Plans finden Sie im [Savings Plans User Guide](#).

Wie funktionieren die Allokationsstrategien mit Gewichten

Wenn Sie den `WeightedCapacity` Parameter in Ihren Überschreibungen (`"DesiredCapacityType": "vcpu"` oder `"DesiredCapacityType": "memory-mib"` auf Gruppenebene) angeben, funktionieren die Zuweisungsstrategien genauso wie bei anderen Auto Scaling Scaling-Gruppen.

Angenommen, Sie haben eine Auto Scaling Scaling-Gruppe mit mehreren Instance-Typen, die unterschiedliche Mengen von v haben CPUs. Sie verwenden `lowest-price` für Ihre Spot- und On-Demand-Zuweisungsstrategien. Wenn Sie sich dafür entscheiden, Gewichtungen auf der Grundlage der vCPU-Anzahl jedes Instance-Typs zuzuweisen, startet Amazon EC2 Auto Scaling die Instance-Typen, die zum Zeitpunkt der Erfüllung den niedrigsten Preis für die von Ihnen zugewiesenen Gewichtungswerte haben (z. B. pro vCPU). Wenn es sich um eine Spot-Instance handelt, dann ist dies der niedrigste Spot-Preis pro vCPU. Wenn es sich um eine On-Demand-Instance handelt, dann ist dies der niedrigste On-Demand-Preis pro vCPU.

Weitere Informationen finden Sie unter [Konfigurieren Sie eine Auto Scaling Scaling-Gruppe für die Verwendung von Instanzgewichten](#).

Gruppe mit gemischten Instanzen mithilfe der attributbasierten Instanztypauswahl erstellen

Anstatt Instance-Typen manuell für Ihre gemischte Instances-Gruppe auszuwählen, können Sie eine Reihe von Instance-Attributen angeben, die Ihre Rechenanforderungen beschreiben. Da Amazon EC2 Auto Scaling Instances startet, müssen alle Instance-Typen, die von der Auto-Scaling-

Gruppe verwendet werden, Ihren gewünschten Instance-Attributen entsprechen. Dies ist bekannt als attributbasierte Instance-Typauswahl.

Dieser Ansatz ist ideal für Workloads und Frameworks, die bei der Wahl der Instance-Typen flexibel sind, wie z.B. Container, Big Data und CI/CD.

Im Folgenden finden Sie die Vorteile der attributbasierten Auswahl von Instance-Typen:

- Optimale Flexibilität für Spot-Instances — Amazon EC2 Auto Scaling kann aus einer Vielzahl von Instance-Typen für den Start von Spot-Instances wählen. Dies entspricht der bewährten Spot-Praxis, bei den Instance-Typen flexibel zu sein, wodurch der Amazon-EC2-Spot-Service eine bessere Chance hat, die von Ihnen benötigte Menge an Rechenkapazität zu finden und zuzuweisen.
- Einfacher Einsatz der richtigen Instance-Typen – Da so viele Instance-Typen verfügbar sind, kann es zeitaufwendig sein, die richtigen Instance-Typen für Ihre Workload zu finden. Wenn Sie Instance-Attribute angeben, haben die Instance-Typen automatisch die erforderlichen Attribute für Ihre Workload.
- Automatische Verwendung neuer Instance-Typen — Ihre Auto Scaling Scaling-Gruppen können Instance-Typen der neueren Generation verwenden, sobald sie veröffentlicht werden. Instance-Typen der neueren Generation werden automatisch verwendet, wenn sie Ihren Anforderungen entsprechen und mit den Zuweisungsstrategien übereinstimmen, die Sie für Ihre Auto-Scaling-Gruppe gewählt haben.

Themen

- [Attributbasierte Auswahl von Instance-Typen](#)
- [Preisschutz](#)
- [Leistungsschutz](#)
- [Voraussetzungen](#)
- [Erstellen Sie eine gemischte Instanzgruppe mit attributbasierter Instanztypauswahl \(Konsole\)](#)
- [Erstellen Sie eine gemischte Instanzgruppe mit einer attributbasierten Instanztypauswahl \(AWS CLI\)](#)
- [Beispielkonfiguration](#)
- [Eine Vorschau Ihrer Instance-Typen anzeigen](#)
- [Zugehörige Ressourcen](#)

Attributbasierte Auswahl von Instance-Typen

Bei der attributbasierten Auswahl von Instance-Typen geben Sie statt einer Liste bestimmter Instance-Typen eine Liste von Instance-Attributen an, die Ihre Instances benötigen, wie z. B.:

- vCPU-Anzahl — Die minimale und maximale Anzahl von v CPUs pro Instanz.
- Arbeitsspeicher — Das Minimum und das Maximum GiBs an Arbeitsspeicher pro Instanz.
- Lokaler Speicher – Ob EBS- oder Instance-Speicher-Volumes für den lokalen Speicher verwendet werden sollen.
- Burstable Performance – Gibt an, ob die T-Instance-Familie verwendet werden soll, einschließlich der Typen T4g, T3a, T3 und T2.

Es stehen viele Optionen zur Definition Ihrer Instanzanforderungen zur Verfügung. Eine Beschreibung der einzelnen Optionen und der Standardwerte finden Sie [InstanceRequirements](#) in der Amazon EC2 Auto Scaling API-Referenz.

Wenn Ihre Auto Scaling Scaling-Gruppe eine Instance starten muss, sucht sie nach Instance-Typen, die Ihren angegebenen Attributen entsprechen und in dieser Availability Zone verfügbar sind. Die Zuweisungsstrategie bestimmt dann, welcher der passenden Instance-Typen gestartet werden soll. Standardmäßig ist für die attributbasierte Instance-Typauswahl eine Preisschutzfunktion aktiviert, um zu verhindern, dass Ihre Auto Scaling Scaling-Gruppe Instance-Typen startet, die Ihre Budgetschwellenwerte überschreiten.

Standardmäßig verwenden Sie die Anzahl der Instances als Maßeinheit, wenn Sie die gewünschte Kapazität Ihrer Auto Scaling Scaling-Gruppe festlegen, was bedeutet, dass jede Instanz als eine Einheit zählt.

Alternativ können Sie den Wert für die gewünschte Kapazität auf die Anzahl von v CPUs oder die Speichermenge festlegen. Verwenden Sie dazu das Dropdown-Feld Gewünschter Kapazitätstyp in der AWS-Managementkonsole oder der `DesiredCapacityType` Eigenschaft in der `UpdateAutoScalingGroup` API-Operation `CreateAutoScalingGroup` oder. Amazon EC2 Auto Scaling startet dann die Anzahl der Instances, die erforderlich sind, um die gewünschte vCPU- oder Speicherkapazität zu erreichen. Wenn Sie beispielsweise v CPUs als gewünschten Kapazitätstyp verwenden und Instances mit CPUs jeweils 2 V verwenden, CPUs würde eine gewünschte Kapazität von 10 v 5 Instances starten. Dies ist eine nützliche Alternative zu [Instance-Gewichten](#).

Preisschutz

Mit Preisschutz können Sie den Höchstpreis angeben, den Sie bereit sind, für EC2-Instances zu zahlen, die von Ihrer Auto Scaling Scaling-Gruppe gestartet wurden. Der Preisschutz ist eine Funktion, die verhindert, dass Ihre Auto Scaling Scaling-Gruppe Instance-Typen verwendet, die Sie für zu teuer halten würden, selbst wenn sie zufällig den von Ihnen angegebenen Attributen entsprechen.

Der Preisschutz ist standardmäßig aktiviert und hat separate Preisschwellen für On-Demand-Instances und Spot-Instances. Wenn Amazon EC2 Auto Scaling neue Instances starten muss, werden Instance-Typen, deren Preis über dem entsprechenden Schwellenwert liegt, nicht gestartet.

Themen

- [Preisschutz auf Abruf](#)
- [Schutz vor Spot-Preisen](#)
- [Passen Sie den Preisschutz individuell an](#)

Preisschutz auf Abruf

Für On-Demand-Instances definieren Sie den maximalen On-Demand-Preis, den Sie zu zahlen bereit sind, als Prozentsatz, der über dem angegebenen On-Demand-Preis liegt. Der identifizierte On-Demand-Preis ist der Preis des günstigsten Instance-Typs C, M oder R der aktuellen Generation mit den von Ihnen angegebenen Attributen.

Wenn ein On-Demand-Preisschutzwert nicht explizit definiert ist, wird ein standardmäßiger maximaler On-Demand-Preis verwendet, der 20 Prozent über dem angegebenen On-Demand-Preis liegt.

Schutz vor Spot-Preisen

Standardmäßig wendet Amazon EC2 Auto Scaling automatisch den optimalen Spot-Instance-Preisschutz an, um konsistent aus einer Vielzahl von Instance-Typen auszuwählen. Sie können den Preisschutz auch manuell selbst festlegen. Wenn Sie dies jedoch Amazon EC2 Auto Scaling für Sie erledigen lassen, können Sie die Wahrscheinlichkeit erhöhen, dass Ihre Spot-Kapazität ausgeschöpft ist.

Sie können den Preisschutz mit einer der folgenden Optionen manuell angeben. Wenn Sie den Preisschutz manuell festlegen, empfehlen wir, die erste Option zu verwenden.

- Ein Prozentsatz eines identifizierten On-Demand-Preises — Der identifizierte On-Demand-Preis ist der Preis des günstigsten Instance-Typs C, M oder R der aktuellen Generation mit Ihren angegebenen Attributen.
- Ein Prozentsatz höher als ein identifizierter Spot-Preis — Der identifizierte Spot-Preis ist der Preis des günstigsten Instance-Typs C, M oder R der aktuellen Generation mit den von Ihnen angegebenen Attributen. Wir empfehlen, diese Option nicht zu verwenden, da die Spot-Preise schwanken können und daher auch Ihr Preisschutzschwellenwert schwanken kann.

Passen Sie den Preisschutz individuell an

Sie können die Schwellenwerte für den Preisschutz in der Amazon EC2 Auto Scaling Scaling-Konsole oder mithilfe von oder anpassen. AWS CLI SDKs

- Verwenden Sie in der Konsole die Einstellungen On-Demand-Preisschutz und Spot-Preisschutz unter Zusätzliche Instance-Attribute.
- Verwenden Sie in der [InstanceRequirements](#) Struktur die `OnDemandMaxPricePercentageOverLowestPrice` Eigenschaft, um den Schwellenwert für den Preisschutz bei On-Demand-Instances anzugeben. Um den Schwellenwert für den Preisschutz für Spot-Instances anzugeben, verwenden Sie entweder die `SpotMaxPricePercentageOverLowestPrice` Eigenschaft `MaxSpotPriceAsPercentageOfOptimalOnDemandPrice` oder.

Wenn Sie den gewünschten Kapazitätstyp (`DesiredCapacityType`) auf v CPUs oder Memory GiB festlegen, gilt der Preisschutz auf der Grundlage des Preises pro vCPU oder pro Speicher und nicht auf dem Preis pro Instance.

Sie können den Preisschutz auch deaktivieren. Um anzugeben, dass es keinen Preisschutzschwellenwert gibt, geben Sie einen hohen Prozentsatz an, z. B. 999999.

Note

Wenn keine Instance-Typen der aktuellen Generation C, M oder R Ihren angegebenen Attributen entsprechen, gilt der Preisschutz trotzdem. Wenn keine Übereinstimmung gefunden wird, stammt der identifizierte Preis von den günstigsten Instance-Typen der aktuellen Generation oder, falls dies nicht der Fall ist, von den günstigsten Instance-Typen der vorherigen Generation, die Ihren Attributen entsprechen.

Leistungsschutz

Der Leistungsschutz ist eine Funktion, die sicherstellt, dass Ihre Auto Scaling Scaling-Gruppe Instance-Typen verwendet, die einer bestimmten Leistungsbasislinie ähnlich sind oder diese übertreffen. Um den Leistungsschutz zu nutzen, geben Sie eine Instance-Familie als Basisreferenz an. Die Funktionen der angegebenen Instance-Familie stellen das niedrigste akzeptable Leistungsniveau dar. Wenn Auto Scaling Instance-Typen auswählt, berücksichtigt es Ihre angegebenen Attribute und die Leistungsbasis. Instance-Typen, die unter die Leistungsbasis fallen, werden automatisch von der Auswahl ausgeschlossen, auch wenn sie Ihren anderen angegebenen Attributen entsprechen. Dadurch wird sichergestellt, dass alle ausgewählten Instance-Typen eine ähnliche oder bessere Leistung als die von der angegebenen Instance-Familie festgelegte Leistungsbasis bieten. Auto Scaling verwendet diese Baseline als Leitfaden für die Auswahl des Instance-Typs, es gibt jedoch keine Garantie dafür, dass die ausgewählten Instance-Typen immer die Baseline für jede Anwendung überschreiten.

Derzeit unterstützt dieses Feature nur die CPU-Leistung als Basisleistungsfaktor. Die CPU-Leistung der angegebenen Instance-Familie dient als Leistungsbasis und stellt sicher, dass die ausgewählten Instance-Typen dieser Basislinie ähnlich sind oder diese übertreffen. Instance-Familien mit denselben CPU-Prozessoren führen zu denselben Filterergebnissen, auch wenn sich ihre Netzwerk- oder Festplattenleistung unterscheidet. Wenn Sie beispielsweise entweder `c6in` oder `c6i` als Basisreferenz angeben, würde dies zu identischen leistungsbasierten Filterergebnissen führen, da beide Instance-Familien denselben CPU-Prozessor verwenden.

Nicht unterstützte Instance-Familien

Die folgenden Instance-Familien werden aus Gründen des Leistungsschutzes nicht unterstützt:

- `c1`
- `g3` | `g3s`
- `hpc7g`
- `m1` | `m2`
- `mac1` | `mac2` | `mac2-m1ultra` | `mac2-m2` | `mac2-m2pro`
- `p3dn` | `p4d` | `p5`
- `t1`
- `u-12tb1` | `u-18tb1` | `u-24tb1` | `u-3tb1` | `u-6tb1` | `u-9tb1` | `u7i-12tb` | `u7in-16tb` | `u7in-24tb` | `u7in-32tb`

Wenn Sie den Leistungsschutz aktivieren, indem Sie eine unterstützte Instance-Familie angeben, schließen die zurückgegebenen Instance-Typen die oben genannten nicht unterstützten Instance-Familien aus.

Beispiel: Einen Basiswert für die CPU-Leistung festlegen

Im folgenden Beispiel besteht die Instance-Anforderung darin, mit Instance-Typen zu starten, deren CPU-Kerne genauso leistungsfähig sind wie die `c6i`-Instance-Familie. Dadurch werden Instance-Typen mit weniger leistungsfähigen CPU-Prozessoren herausgefiltert, auch wenn sie Ihre anderen angegebenen Instance-Anforderungen erfüllen, z. B. die Anzahl von V. CPUs Wenn Ihre angegebenen Instance-Attribute beispielsweise 4 V CPUs und 16 GB Arbeitsspeicher beinhalten, `c6i` wird ein Instance-Typ mit diesen Attributen, aber mit einer geringeren CPU-Leistung als dieser, von der Auswahl ausgeschlossen.

```
"BaselinePerformanceFactors": {
  "Cpu": {
    "References": [
      {
        "InstanceFamily": "c6i"
      }
    ]
  }
}
```

Überlegungen

Beachten Sie bei der Verwendung von Performance Protection Folgendes:

- Sie können entweder Instance-Typen oder Instance-Attribute angeben, aber nicht beide gleichzeitig.
- Sie können maximal vier `InstanceRequirements`-Strukturen in einer Anforderungskonfiguration angeben.

Voraussetzungen

- Erstellen Sie eine Startvorlage. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).
- Stellen Sie sicher, dass die Startvorlage nicht bereits Spot-Instances anfordert.

Erstellen Sie eine gemischte Instanzgruppe mit attributbasierter Instanztypauswahl (Konsole)

Gehen Sie wie folgt vor, um eine gemischte Instances-Gruppe zu erstellen, indem Sie die attributbasierte Auswahl des Instance-Typs verwenden. Um die Schritte effizient ausführen zu können, wurden einige optionale Abschnitte übersprungen.

Für die meisten Allzweck-Workloads reicht es aus, die Anzahl von V CPUs und den Arbeitsspeicher anzugeben, die Sie benötigen. Für fortgeschrittene Anwendungsfälle können Sie Attribute wie Speichertyp, Netzwerkschnittstellen, CPU-Hersteller und Beschleunigertyp angeben.

Eine Übersicht der bewährten Methoden für eine Gruppe mit gemischten Instanzen finden Sie unter [Übersicht über die Einrichtung für die Erstellung einer gemischten Instance-Gruppe](#).

So erstellen Sie eine gemischte Instances-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie auf der Navigationsleiste oben auf dem Bildschirm dieselbe AWS-Region, die Sie bei der Erstellung der Startvorlage angegeben haben.
3. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
4. Geben Sie auf der Seite Startvorlage oder -konfiguration auswählen für Auto-Scaling-Gruppenname einen Namen für Ihre Auto-Scaling-Gruppe ein.
5. Gehen Sie folgendermaßen vor, um Ihre Startungsvorlage auszuwählen:
 - a. Wählen Sie für Launch template (Startvorlage) eine vorhandene Startvorlage aus.
 - b. Wählen Sie unter Launch template version (Version der Startvorlage) aus, ob die Auto-Scaling-Gruppe beim horizontalen Skalieren nach oben die standardmäßige, die neueste oder eine bestimmte Version der Startvorlage verwenden soll.
 - c. Stellen Sie sicher, dass Ihre Startvorlage alle Optionen unterstützt, die Sie verwenden möchten, und wählen Sie dann Next (Weiter) aus.
6. Wählen Sie auf der Seite Instance-Startoptionen auswählen die folgenden Einstellungen aus.
 - a. Wählen Sie für Instance-Typanforderungen die Option Startvorlage überschreiben.

Note

Bei Auswahl einer Startvorlage, die bereits eine Reihe von Instance-Attributen wie etwa vCPUs und Arbeitsspeicher enthält, werden die Instance-Attribute angezeigt.

Diese Attribute werden zu den Eigenschaften der Auto-Scaling-Gruppe hinzugefügt, wo Sie sie jederzeit über die Amazon EC2 Auto Scaling-Konsole aktualisieren können.

- b. Geben Sie unter Instance-Attribute angeben zunächst Ihre V CPUs - und Speicheranforderungen ein.
 - Geben Sie für v CPUs die gewünschte Mindest- und Höchstzahl von v einCPUs. Um kein Limit anzugeben, wählen Sie Kein Minimum, Kein Maximum oder beides aus.
 - Geben Sie für Arbeitsspeicher (GiB) den gewünschten Mindest- und Höchstwert ein. Um kein Limit anzugeben, wählen Sie Kein Minimum, Kein Maximum oder beide Optionen aus.
- c. (Optional) Für Zusätzliche Instance-Attribute können Sie optional ein oder mehrere Attribute angeben, um Ihre Computinganforderungen genauer auszudrücken. Jedes zusätzliche Attribut fügt Ihrer Anfrage weitere Einschränkungen hinzu.
- d. Erweitern Sie Vorschau der passenden Instance-Typen, um die Instance-Typen mit Ihren angegebenen Attributen anzuzeigen.
- e. Geben Sie unter Optionen für den Kauf von Instances unter Instances Distribution die Prozentsätze der Gruppe an, die als On-Demand-Instances bzw. Spot Instances gestartet werden sollen. Wenn Ihre Anwendung zustandslos und fehlertolerant ist und damit umgehen kann, dass eine Instance unterbrochen wird, können Sie einen höheren Prozentsatz an Spot-Instances angeben.
- f. (Optional) Wenn Sie sich für einen Prozentsatz an Spot Instances entschieden haben, können Sie On-Demand-Basiskapazität einbeziehen auswählen und dann die Mindestmenge der Anfangskapazität der Auto-Scaling-Gruppe angeben, die von On-Demand-Instances erfüllt werden muss. Für alles, was über die Basiskapazität hinausgeht, werden die Einstellungen für die Instance-Verteilung verwendet, um zu bestimmen, wie viele On-Demand-Instances und Spot-Instances gestartet werden sollen.
- g. Unter Zuteilungsstrategien wird für die On-Demand-Zuteilungsstrategie automatisch der niedrigste Preis ausgewählt und kann nicht geändert werden.
- h. Wählen Sie für die Spot-Zuweisungsstrategie eine Zuweisungsstrategie. Price capacity optimized (Preiskapazität optimiert) ist standardmäßig ausgewählt.
- i. Für Kapazitätsausgleich wählen Sie aus, ob Sie den Kapazitätsausgleich aktivieren oder deaktivieren möchten. Verwenden Sie Capacity Rebalancing, um automatisch zu reagieren, wenn Ihre Spot Instances aufgrund einer Spot-Unterbrechung bald beendet werden.

- Weitere Informationen finden Sie unter [Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances](#).
- j. Wählen Sie unter Netzwerk für VPC eine VPC. Die Auto-Scaling-Gruppe muss in derselben VPC erstellt werden wie die Sicherheitsgruppe, die Sie in Ihrer Startvorlage angegeben haben.
 - k. Wählen Sie für Availability Zones and subnets (Subnetz) eines der öffentlichen Subnetze in der festgelegten VPC aus. Verwenden Sie Subnetze in mehreren Availability Zones, um eine hohe Verfügbarkeit zu erzielen. Weitere Informationen finden Sie unter [Überlegungen bei der Auswahl von VPC-Subnetzen](#).
 - l. Wählen Sie Weiter, Weiter aus.
7. Gehen Sie für den Schritt Gruppengröße und Skalierungsrichtlinien konfigurieren wie folgt vor:
- a. Um Ihre gewünschte Kapazität in anderen Einheiten als Instances zu messen, wählen Sie die entsprechende Option für Gruppengröße, Gewünschter Kapazitätstyp aus. Einheiten CPUs, v und Memory GiB werden unterstützt. Standardmäßig gibt Amazon EC2 Auto Scaling Einheiten an, was sich in einer Anzahl von Instances niederschlägt.
 - b. Für Gewünschte Kapazität, die Anfangsgröße Ihrer Auto-Scaling-Gruppe.
 - c. Wenn im Abschnitt Skalierung unter Skalierungslimits Ihr neuer Wert für die gewünschte Kapazität größer als die gewünschte Mindestkapazität und die gewünschte Höchstkapazität ist, wird die gewünschte Höchstkapazität automatisch auf den neuen Wert für die gewünschte Kapazität erhöht. Sie können die Limits bei Bedarf ändern. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).
8. Wählen Sie Skip to review (Mit Prüfen fortfahren) aus.
9. Wählen Sie auf der Seite Review (Prüfen) Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Erstellen Sie eine gemischte Instanzgruppe mit einer attributbasierten Instanztypauswahl ()AWS CLI

Erstellen Sie wie folgt eine gemischte Instances-Gruppe über die Befehlszeile:

Verwenden Sie einen der folgenden Befehle:

- [create-auto-scaling-group](#) (AWS CLI)
- [Neu](#) - () ASAuto ScalingGroup AWS Tools for Windows PowerShell

Beispielkonfiguration

Verwenden Sie den folgenden Befehl, um eine Auto Scaling Scaling-Gruppe mit attributbasierter Instanztypauswahl mithilfe von zu erstellen. AWS CLI [create-auto-scaling-group](#)

Die folgenden Instance-Attribute werden angegeben:

- **VCpuCount**— Die Instance-Typen müssen mindestens vier V CPUs und maximal acht V haben. CPUs
- **MemoryMiB** – Die Instance-Typen müssen über mindestens 16.384 MiB Arbeitsspeicher verfügen.
- **CpuManufacturers** – Die Instance-Typen müssen eine von Intel hergestellte CPU haben.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Im Folgenden sehen Sie ein Beispiel für eine `config.json`-Datei.

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredCapacityType": "units",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "Default"
      },
    },
    "Overrides": [{
      "InstanceRequirements": {
        "VCpuCount": {"Min": 4, "Max": 8},
        "MemoryMiB": {"Min": 16384},
        "CpuManufacturers": ["intel"]
      }
    }]
  },
  "InstancesDistribution": {
    "OnDemandPercentageAboveBaseCapacity": 50,
    "SpotAllocationStrategy": "price-capacity-optimized"
  }
},
```

```
"MinSize": 0,  
"MaxSize": 100,  
"DesiredCapacity": 4,  
"DesiredCapacityType": "units",  
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"  
}
```

Um den Wert für die gewünschte Kapazität als die Anzahl von v CPUs oder die Speichermenge festzulegen, geben Sie "DesiredCapacityType": "vcpu" oder "DesiredCapacityType": "memory-mib" in der Datei an. Der Standardtyp für die gewünschte Kapazität ist units, wodurch der Wert für die gewünschte Kapazität als Anzahl der Instanzen festgelegt wird.

YAML

Alternativ können Sie den folgenden [create-auto-scaling-group](#) Befehl verwenden, um die Auto Scaling Scaling-Gruppe zu erstellen. Dadurch wird auf eine YAML-Datei als einziger Parameter für Ihre Auto-Scaling-Gruppe verwiesen.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

Im Folgenden sehen Sie ein Beispiel für eine config.yaml-Datei.

```
---  
AutoScalingGroupName: my-asg  
DesiredCapacityType: units  
MixedInstancesPolicy:  
  LaunchTemplate:  
    LaunchTemplateSpecification:  
      LaunchTemplateName: my-launch-template  
      Version: $Default  
    Overrides:  
      - InstanceRequirements:  
        VCpuCount:  
          Min: 2  
          Max: 4  
        MemoryMiB:  
          Min: 2048  
        CpuManufacturers:  
          - intel  
  InstancesDistribution:  
    OnDemandPercentageAboveBaseCapacity: 50
```

```
SpotAllocationStrategy: price-capacity-optimized
MinSize: 0
MaxSize: 100
DesiredCapacity: 4
DesiredCapacityType: units
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Um den Wert für die gewünschte Kapazität als die Anzahl von v CPUs oder die Speichermenge festzulegen, geben Sie `DesiredCapacityType: vcpu` oder `DesiredCapacityType: memory-mib` in der Datei an. Der Standardtyp für die gewünschte Kapazität ist `units`, wodurch der Wert für die gewünschte Kapazität als Anzahl der Instanzen festgelegt wird.

Ein Beispiel für die Verwendung mehrerer Startvorlagen mit attributbasierter Instance-Typauswahl finden Sie unter [Verwenden von mehreren Startvorlagen](#)

Eine Vorschau Ihrer Instance-Typen anzeigen

Sie können die Instance-Typen, die Ihren Rechenanforderungen entsprechen, in der Vorschau anzeigen, ohne sie zu starten, und Ihre Anforderungen bei Bedarf anpassen. Wenn Sie Ihre Auto-Scaling-Gruppe in der Amazon EC2 Auto Scaling-Konsole erstellen, erscheint eine Vorschau der Instance-Typen im Abschnitt `Preview matching instance types` (Vorschau passender Instance-Typen) auf der Seite `Choose instance launch options` (Startoptionen für Instances auswählen).

Alternativ können Sie eine Vorschau der Instance-Typen anzeigen, indem Sie einen Amazon EC2 [GetInstanceTypesFromInstanceRequirements](#) EC2-API-Aufruf mit dem AWS CLI oder einem SDK durchführen. Übergeben Sie die `InstanceRequirements`-Parameter in der Anfrage genau in dem Format, das Sie zum Erstellen oder Aktualisieren einer Auto-Scaling-Gruppe verwenden würden. Weitere Informationen finden Sie unter [Preview-Instance-Typen mit bestimmten Attributen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Zugehörige Ressourcen

Weitere Informationen zur attributbasierten Instanztypauswahl finden Sie unter [Attributbasierte Instanztypauswahl für EC2 Auto Scaling und EC2 Fleet](#) im Blog. AWS

Sie können eine attributbasierte Auswahl des Instance-Typs erklären, wenn Sie eine Auto-Scaling-Gruppe mit CloudFormation erstellen. Weitere Informationen finden Sie unter [Auto Scaling-Vorlagenbeispiele](#) im Abschnitt des CloudFormation -Benutzerhandbuchs.

Erstellen Sie eine Gruppe mit gemischten Instances, indem Sie die Instance-Typen manuell auswählen

In diesem Thema erfahren Sie, wie Sie mehrere Instance-Typen in einer einzelnen Auto-Scaling-Gruppe starten, indem Sie die Instance-Typen manuell auswählen.

Wenn Sie Instance-Attribute lieber als Kriterien für die Auswahl von Instance-Typen verwenden möchten, finden Sie weitere Informationen unter [Gruppe mit gemischten Instanzen mithilfe der attributbasierten Instanztypauswahl erstellen](#).

Inhalt

- [Voraussetzungen](#)
- [Eine gemischte Instances-Gruppe \(Konsole\) erstellen](#)
- [Eine gemischte Instances-Gruppe \(AWS CLI\) erstellen](#)
- [Beispielkonfigurationen](#)

Voraussetzungen

- Erstellen Sie eine Startvorlage. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).
- Stellen Sie sicher, dass die Startvorlage nicht bereits Spot-Instances anfordert.

Eine gemischte Instances-Gruppe (Konsole) erstellen

Gehen Sie wie folgt vor, um eine Instances-Gruppe zu erstellen, indem Sie manuell auswählen, welche Instance-Typen Ihre Gruppe starten kann. Um die Schritte effizient ausführen zu können, wurden einige optionale Abschnitte übersprungen.

Eine Übersicht der bewährten Methoden für eine Gruppe mit gemischten Instanzen finden Sie unter [Übersicht über die Einrichtung für die Erstellung einer gemischten Instance-Gruppe](#).

So erstellen Sie eine gemischte Instances-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie auf der Navigationsleiste oben auf dem Bildschirm dieselbe AWS-Region, die Sie bei der Erstellung der Startvorlage angegeben haben.

3. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
4. Geben Sie auf der Seite Startvorlage oder -konfiguration auswählen für Auto-Scaling-Gruppenname einen Namen für Ihre Auto-Scaling-Gruppe ein.
5. Gehen Sie folgendermaßen vor, um Ihre Startungsvorlage auszuwählen:
 - a. Wählen Sie für Launch template (Startvorlage) eine vorhandene Startvorlage aus.
 - b. Wählen Sie unter Launch template version (Version der Startvorlage) aus, ob die Auto-Scaling-Gruppe beim horizontalen Skalieren nach oben die standardmäßige, die neueste oder eine bestimmte Version der Startvorlage verwenden soll.
 - c. Stellen Sie sicher, dass Ihre Startvorlage alle Optionen unterstützt, die Sie verwenden möchten, und wählen Sie dann Next (Weiter) aus.
6. Wählen Sie auf der Seite Instance-Startoptionen auswählen die folgenden Einstellungen aus.
 - a. Wählen Sie für Instance type requirements (Anforderungen an Instance-Typen) Override launch template (Startvorlage überschreiben), Manually add instance types (Startvorlage überschreiben) aus.
 - b. Wählen Sie Ihre Instance-Typen aus. Sie können unsere Empfehlungen als Ausgangspunkt verwenden. Die Option Family and generation flexible (Familie und Generation flexibel) ist standardmäßig ausgewählt.
 - Um die Reihenfolge der Instance-Typen zu ändern, verwenden Sie die Pfeile. Wenn Sie eine Zuweisungsstrategie auswählen, die Priorisierung unterstützt, legt die Reihenfolge der Instance-Typen deren Startpriorität fest.
 - Um einen Instance-Typ zu entfernen, wählen Sie X aus.
 - (Optional) Für die Felder in der Spalte Gewichtung können Sie jedem Instance-Typ eine relative Gewichtung zuweisen. Geben Sie dazu die Anzahl der Einheiten ein, die eine Instance dieses Typs zur gewünschten Kapazität der Gruppe beiträgt. Dies kann nützlich sein, wenn sich bei den Instance-Typen die vCPU, der Arbeitsspeicher, der Speicherplatz oder die Netzwerkbandbreitenfunktionen unterscheiden. Weitere Informationen finden Sie unter [Konfigurieren Sie eine Auto Scaling Scaling-Gruppe für die Verwendung von Instanzgewichten](#).

Wenn Sie sich für Empfehlungen vom Typ Größe flexibel entschieden haben, haben alle Instance-Typen, die Teil dieses Abschnitts sind, automatisch einen Gewichtungswert. Wenn Sie keine Gewichtungen angeben möchten, leeren Sie die Felder in der Spalte Weight (Gewichtung) für alle Instance-Typen.

- c. Geben Sie unter Optionen für den Kauf von Instances unter Instances Distribution die Prozentsätze der Gruppe an, die als On-Demand-Instances bzw. Spot-Instances gestartet werden sollen. Wenn Ihre Anwendung zustandslos und fehlertolerant ist und damit umgehen kann, dass eine Instance unterbrochen wird, können Sie einen höheren Prozentsatz an Spot-Instances angeben.
- d. (Optional) Wenn Sie sich für einen Prozentsatz an Spot Instances entschieden haben, können Sie On-Demand-Basiskapazität einbeziehen auswählen und dann die Mindestmenge der Anfangskapazität der Auto-Scaling-Gruppe angeben, die von On-Demand-Instances erfüllt werden muss. Für alles, was über die Basiskapazität hinausgeht, werden die Einstellungen für die Instance-Verteilung verwendet, um zu bestimmen, wie viele On-Demand-Instances und Spot-Instances gestartet werden sollen.
- e. Wählen Sie unter Zuteilungsstrategien für On-Demand-Zuteilungsstrategie eine Zuteilungsstrategie. Wenn Sie Ihre Instance-Typen manuell auswählen, ist Prioritized (Priorisiert) standardmäßig ausgewählt.
- f. Wählen Sie für die Spot-Zuweisungsstrategie eine Zuweisungsstrategie. Price capacity optimized (Preiskapazität optimiert) ist standardmäßig ausgewählt.

Wenn Sie Kapazitätsoptimiert ausgewählt haben, können Sie optional das Feld Instance-Typen priorisieren aktivieren, um Amazon EC2 Auto Scaling auf der Grundlage der Reihenfolge, in der Ihre Instance-Typen aufgeführt sind, auswählen zu lassen, welcher Instance-Typ zuerst gestartet wird.

- g. Für Kapazitätsausgleich wählen Sie aus, ob Sie den Kapazitätsausgleich aktivieren oder deaktivieren möchten. Verwenden Sie Capacity Rebalancing, um automatisch zu reagieren, wenn Ihre Spot Instances aufgrund einer Spot-Unterbrechung bald beendet werden. Weitere Informationen finden Sie unter [Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances](#).
 - h. Wählen Sie unter Netzwerk für VPC eine VPC. Die Auto-Scaling-Gruppe muss in derselben VPC erstellt werden wie die Sicherheitsgruppe, die Sie in Ihrer Startvorlage angegeben haben.
 - i. Wählen Sie für Availability Zones and subnets (Subnetz) eines der öffentlichen Subnetze in der festgelegten VPC aus. Verwenden Sie Subnetze in mehreren Availability Zones, um eine hohe Verfügbarkeit zu erzielen. Weitere Informationen finden Sie unter [Überlegungen bei der Auswahl von VPC-Subnetzen](#).
 - j. Wählen Sie Weiter, Weiter aus.
7. Gehen Sie für den Schritt Gruppengröße und Skalierungsrichtlinien konfigurieren wie folgt vor:

- a. Geben Sie unter Gruppengröße für Gewünschte Kapazität die anfängliche Anzahl von Instances ein, die gestartet werden sollen.

Standardmäßig wird die gewünschte Kapazität als Anzahl von Instances ausgedrückt. Wenn Sie Ihren Instance-Typen Gewichtungen zugewiesen haben, müssen Sie diesen Wert in dieselbe Maßeinheit umrechnen, die Sie für die Zuweisung von Gewichtungen verwendet haben, z. B. in die Anzahl von v. CPUs

- b. Wenn im Abschnitt Skalierung unter Skalierungslimits Ihr neuer Wert für die gewünschte Kapazität größer als die gewünschte Mindestkapazität und die gewünschte Höchstkapazität ist, wird die gewünschte Höchstkapazität automatisch auf den neuen Wert für die gewünschte Kapazität erhöht. Sie können die Limits bei Bedarf ändern. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).

8. Wählen Sie Skip to review (Mit Prüfen fortfahren) aus.
9. Wählen Sie auf der Seite Review (Prüfen) Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Eine gemischte Instances-Gruppe (AWS CLI) erstellen

Erstellen Sie wie folgt eine gemischte Instances-Gruppe über die Befehlszeile:

Verwenden Sie einen der folgenden Befehle:

- [create-auto-scaling-group](#) (AWS CLI)
- [Neu- ASAuto ScalingGroup](#) (AWS Tools for Windows PowerShell)

Beispielkonfigurationen

Die folgenden Beispielkonfigurationen zeigen, wie gemischte Instance-Gruppen mit verschiedenen Spot-Zuweisungsstrategien erstellt werden können.

Note

Diese Beispiele zeigen, wie Sie eine Konfigurationsdatei verwenden, die in JSON oder YAML formatiert ist. Wenn Sie AWS CLI Version 1 verwenden, müssen Sie eine Konfigurationsdatei

im JSON-Format angeben. Wenn Sie AWS CLI Version 2 verwenden, können Sie eine Konfigurationsdatei angeben, die entweder in YAML oder JSON formatiert ist.

Beispiele

- [Beispiel 1: Starten von Spot-Instances mit der capacity-optimized- Zuweisungsstrategie](#)
- [Beispiel 2: Starten von Spot-Instances mit der capacity-optimized-prioritized- Zuweisungsstrategie](#)
- [Beispiel 3: Starten von Spot-Instances mit der über zwei Pools diversifizierten lowest-price- Zuweisungsstrategie](#)
- [Beispiel 4: Starten von Spot-Instances mit der price-capacity-optimized- Zuweisungsstrategie](#)

Beispiel 1: Starten von Spot-Instances mit der **capacity-optimized**- Zuweisungsstrategie

Mit dem folgenden [create-auto-scaling-group](#) Befehl wird eine Auto Scaling Scaling-Gruppe erstellt, die Folgendes spezifiziert:

- Der Prozentsatz der Gruppe, die als On-Demand-Instances gestartet werden soll (0) und eine Basisanzahl von On-Demand-Instances, mit denen begonnen werden soll (1).
- Die in der Prioritätsreihenfolge zu startenden Instance-Typen (c5.large, c5a.large, m5.large, m5a.large, c4.large, m4.large, c3.large, m3.large).
- Die Subnetze, in denen die Instances gestartet werden sollen (subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782). Diese entsprechen jeweils einer anderen Availability Zone.
- Beschreibt eine Startvorlage (my-launch-template) und die Version der Startvorlage (\$Default).

Wenn Amazon EC2 Auto Scaling versucht, Ihre On-Demand-Kapazität zu erfüllen, wird zuerst der c5.large-Instance-Typ gestartet. Die Spot-Instances stammen aus dem optimalen Spot-Pool in jeder Availability Zone basierend auf der Spot-Instance-Kapazität.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Die Datei config.json enthält den folgenden Inhalt.

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Default"
      },
      "Overrides": [
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        },
        {
          "InstanceType": "m5.large"
        },
        {
          "InstanceType": "m5a.large"
        },
        {
          "InstanceType": "c4.large"
        },
        {
          "InstanceType": "m4.large"
        },
        {
          "InstanceType": "c3.large"
        },
        {
          "InstanceType": "m3.large"
        }
      ]
    },
    "InstancesDistribution": {
      "OnDemandBaseCapacity": 1,
      "OnDemandPercentageAboveBaseCapacity": 0,
      "SpotAllocationStrategy": "capacity-optimized"
    }
  },
  "MinSize": 1,
  "MaxSize": 5,
}
```

```
"DesiredCapacity": 3,  
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"  
}
```

YAML

Alternativ können Sie den folgenden [create-auto-scaling-group](#) Befehl verwenden, um die Auto Scaling Group zu erstellen. Dadurch wird auf eine YAML-Datei als einziger Parameter für Ihre Auto-Scaling-Gruppe verwiesen.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

Die Datei `config.yaml` enthält den folgenden Inhalt.

```
---  
AutoScalingGroupName: my-asg  
MixedInstancesPolicy:  
  LaunchTemplate:  
    LaunchTemplateSpecification:  
      LaunchTemplateName: my-launch-template  
      Version: $Default  
    Overrides:  
      - InstanceType: c5.large  
      - InstanceType: c5a.large  
      - InstanceType: m5.large  
      - InstanceType: m5a.large  
      - InstanceType: c4.large  
      - InstanceType: m4.large  
      - InstanceType: c3.large  
      - InstanceType: m3.large  
    InstancesDistribution:  
      OnDemandBaseCapacity: 1  
      OnDemandPercentageAboveBaseCapacity: 0  
      SpotAllocationStrategy: capacity-optimized  
  MinSize: 1  
  MaxSize: 5  
  DesiredCapacity: 3  
  VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Beispiel 2: Starten von Spot-Instances mit der **capacity-optimized-prioritized**-Zuweisungsstrategie

Mit dem folgenden [create-auto-scaling-group](#) Befehl wird eine Auto Scaling Group erstellt, die Folgendes spezifiziert:

- Der Prozentsatz der Gruppe, die als On-Demand-Instances gestartet werden soll (0) und eine Basisanzahl von On-Demand-Instances, mit denen begonnen werden soll (1).
- Die in der Prioritätsreihenfolge zu startenden Instance-Typen (c5.large, c5a.large, m5.large, m5a.large, c4.large, m4.large, c3.large, m3.large).
- Die Subnetze, in denen die Instances gestartet werden sollen (subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782). Diese entsprechen jeweils einer anderen Availability Zone.
- Beschreibt eine Startvorlage (my-launch-template) und die Version der Startvorlage (\$Latest).

Wenn Amazon EC2 Auto Scaling versucht, Ihre On-Demand-Kapazität zu erfüllen, wird zuerst der c5.large-Instance-Typ gestartet. Wenn Amazon EC2 Auto Scaling versucht, Ihre Spot-Kapazität zu erfüllen, erfüllt es die Prioritäten des Instance-Typen auf Best-Effort-Basis. An erster Stelle steht jedoch immer die Kapazitätsoptimierung.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Die Datei config.json enthält den folgenden Inhalt.

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
    },
    "Overrides": [
      {
        "InstanceType": "c5.large"
      },
    ],
  },
}
```

```

    {
      "InstanceType": "c5a.large"
    },
    {
      "InstanceType": "m5.large"
    },
    {
      "InstanceType": "m5a.large"
    },
    {
      "InstanceType": "c4.large"
    },
    {
      "InstanceType": "m4.large"
    },
    {
      "InstanceType": "c3.large"
    },
    {
      "InstanceType": "m3.large"
    }
  ]
},
"InstancesDistribution": {
  "OnDemandBaseCapacity": 1,
  "OnDemandPercentageAboveBaseCapacity": 0,
  "SpotAllocationStrategy": "capacity-optimized-prioritized"
}
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```

YAML

Alternativ können Sie den folgenden [create-auto-scaling-group](#) Befehl verwenden, um die Auto Scaling Gruppe zu erstellen. Dadurch wird auf eine YAML-Datei als einziger Parameter für Ihre Auto-Scaling-Gruppe verwiesen.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

Die Datei `config.yaml` enthält den folgenden Inhalt.

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large
      - InstanceType: c3.large
      - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized-prioritized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Beispiel 3: Starten von Spot-Instances mit der über zwei Pools diversifizierten **lowest-price-**Zuweisungsstrategie

Mit dem folgenden [create-auto-scaling-group](#) Befehl wird eine Auto Scaling Group erstellt, die Folgendes spezifiziert:

- Der Prozentsatz der Gruppe, die als On-Demand-Instances gestartet werden soll (50). (Gibt keine Basisanzahl von On-Demand-Instances an, mit der gestartet werden soll.)
- Die in der Prioritätsreihenfolge zu startenden Instance-Typen (`c5.large`, `c5a.large`, `m5.large`, `m5a.large`, `c4.large`, `m4.large`, `c3.large`, `m3.large`).
- Die Subnetze, in denen die Instances gestartet werden sollen (`subnet-5ea0c127`, `subnet-6194ea3b`, `subnet-c934b782`). Diese entsprechen jeweils einer anderen Availability Zone.

- Beschreibt eine Startvorlage (`my-launch-template`) und die Version der Startvorlage (`$Latest`).

Wenn Amazon EC2 Auto Scaling versucht, Ihre On-Demand-Kapazität zu erfüllen, wird zuerst der `c5.large`-Instance-Typ gestartet. Für Ihre Spot-Kapazität versucht Amazon EC2 Auto Scaling, die Spot Instances gleichmäßig über die beiden kostengünstigsten Pools in jeder Availability Zone zu starten.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Die Datei `config.json` enthält den folgenden Inhalt.

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
      "Overrides": [
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        },
        {
          "InstanceType": "m5.large"
        },
        {
          "InstanceType": "m5a.large"
        },
        {
          "InstanceType": "c4.large"
        },
        {
          "InstanceType": "m4.large"
        },
        {
```

```

        "InstanceType": "c3.large"
      },
      {
        "InstanceType": "m3.large"
      }
    ]
  },
  "InstancesDistribution": {
    "OnDemandPercentageAboveBaseCapacity": 50,
    "SpotAllocationStrategy": "lowest-price",
    "SpotInstancePools": 2
  }
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```

YAML

Alternativ können Sie den folgenden [create-auto-scaling-group](#) Befehl verwenden, um die Auto Scaling Group zu erstellen. Dadurch wird auf eine YAML-Datei als einziger Parameter für Ihre Auto-Scaling-Gruppe verwiesen.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

Die Datei `config.yaml` enthält den folgenden Inhalt.

```

---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large

```

```

- InstanceType: c3.large
- InstanceType: m3.large
InstancesDistribution:
  OnDemandPercentageAboveBaseCapacity: 50
  SpotAllocationStrategy: lowest-price
  SpotInstancePools: 2
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782

```

Beispiel 4: Starten von Spot-Instances mit der **price-capacity-optimized**- Zuweisungsstrategie

Mit dem folgenden [create-auto-scaling-group](#) Befehl wird eine Auto Scaling Scaling-Gruppe erstellt, die Folgendes spezifiziert:

- Der Prozentsatz der Gruppe, die als On-Demand-Instances gestartet werden soll (30). (Gibt keine Basisanzahl von On-Demand-Instances an, mit der gestartet werden soll.)
- Die in der Prioritätsreihenfolge zu startenden Instance-Typen (c5.large, c5a.large, m5.large, m5a.large, c4.large, m4.large, c3.large, m3.large).
- Die Subnetze, in denen die Instances gestartet werden sollen (subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782). Diese entsprechen jeweils einer anderen Availability Zone.
- Beschreibt eine Startvorlage (my-launch-template) und die Version der Startvorlage (\$Latest).

Wenn Amazon EC2 Auto Scaling versucht, Ihre On-Demand-Kapazität zu erfüllen, wird zuerst der c5.large-Instance-Typ gestartet. Für Ihre Spot-Kapazität versucht Amazon EC2 Auto Scaling, die Spot Instances gleichmäßig über Spot-Instance-Pools mit dem günstigsten Preis sowie mit optimaler Kapazität für die Anzahl der zu startenden Instances zu starten.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file:///~/config.json
```

Die Datei config.json enthält den folgenden Inhalt.

```
{
  "AutoScalingGroupName": "my-asg",
```

```
"MixedInstancesPolicy": {
  "LaunchTemplate": {
    "LaunchTemplateSpecification": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "$Latest"
    },
    "Overrides": [
      {
        "InstanceType": "c5.large"
      },
      {
        "InstanceType": "c5a.large"
      },
      {
        "InstanceType": "m5.large"
      },
      {
        "InstanceType": "m5a.large"
      },
      {
        "InstanceType": "c4.large"
      },
      {
        "InstanceType": "m4.large"
      },
      {
        "InstanceType": "c3.large"
      },
      {
        "InstanceType": "m3.large"
      }
    ]
  },
  "InstancesDistribution": {
    "OnDemandPercentageAboveBaseCapacity": 30,
    "SpotAllocationStrategy": "price-capacity-optimized"
  }
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

Alternativ können Sie den folgenden [create-auto-scaling-group](#) Befehl verwenden, um die Auto Scaling Scaling-Gruppe zu erstellen. Dadurch wird auf eine YAML-Datei als einziger Parameter für Ihre Auto-Scaling-Gruppe verwiesen.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

Die Datei `config.yaml` enthält den folgenden Inhalt.

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large
      - InstanceType: c3.large
      - InstanceType: m3.large
    InstancesDistribution:
      OnDemandPercentageAboveBaseCapacity: 30
      SpotAllocationStrategy: price-capacity-optimized
  MinSize: 1
  MaxSize: 5
  DesiredCapacity: 3
  VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Konfigurieren Sie eine Auto Scaling Scaling-Gruppe für die Verwendung von Instanzgewichten

Wenn Sie mehrere Instance-Typen verwenden, können Sie angeben, wie viele Einheiten jedem Instance-Typ zugeordnet werden sollen, und dann die Kapazität Ihrer Gruppe mit derselben Maßeinheit angeben. Diese Option zur Kapazitätsspezifikation wird als Gewichte bezeichnet.

Nehmen wir zum Beispiel an, Sie führen eine rechenintensive Anwendung aus, die mit mindestens 8 vCPUs und 15 GiB RAM am besten funktioniert. Wenn Sie `c5.2xlarge` als Basiseinheit verwenden, würde jeder der folgenden EC2-Instance-Typen Ihre Anwendungsanforderungen erfüllen.

Beispiel für Instance-Typen

Instance-Typ	vCPU	Arbeitsspeicher (GiB)
<code>c5.2xlarge</code>	8	16
<code>c5.4xlarge</code>	16	32
<code>c5.12xlarge</code>	48	96
<code>c5.18xlarge</code>	72	144
<code>c5.24xlarge</code>	96	192

Standardmäßig haben alle Instance-Typen unabhängig von ihrer Größe das gleiche Gewicht. Mit anderen Worten: Unabhängig davon, ob Amazon EC2 Auto Scaling einen großen oder kleinen Instance-Typ startet, zählt jede Instance gleich viel für die gewünschte Kapazität der Auto-Scaling-Gruppe.

Bei Gewichtungen weisen Sie jedoch einen Zahlenwert zu, der angibt, wie viele Einheiten jedem Instance-Typ zugeordnet werden sollen. Wenn die Instances beispielsweise unterschiedliche Größen aufweisen, kann eine `c5.2xlarge`-Instance eine Gewichtung von „2“ haben, und eine (doppelt so große) `c5.4xlarge` könnte eine Gewichtung von „4“ haben usw. Wenn Amazon EC2 Auto Scaling die Gruppe skaliert, geben die Gewichtungen die Anzahl der Einheiten an, die jede Instance auf die gewünschte Kapazität angerechnet wird.

Die Gewichtungen ändern nicht, welche Instance-Typen Amazon EC2 Auto Scaling startet. Stattdessen tun dies die Zuweisungsstrategien. Weitere Informationen finden Sie unter [Zuweisungsstrategien für mehrere Instance-Typen](#).

Wichtig

Um eine Auto Scaling Gruppe so zu konfigurieren, dass sie mithilfe der Anzahl von vCPUs oder der Speichermenge jedes Instance-Typs die gewünschte Kapazität erreicht, empfehlen wir die attributbasierte Instance-Typauswahl. Durch die Einstellung des

`DesiredCapacityType` Parameters wird automatisch die Anzahl der Einheiten angegeben, die jedem Instance-Typ zugeordnet werden sollen, basierend auf dem Wert, den Sie für diesen Parameter festlegen. Weitere Informationen finden Sie unter [Gruppe mit gemischten Instanzen mithilfe der attributbasierten Instanztypauswahl erstellen](#).

Inhalt

- [Überlegungen](#)
- [Verhalten beim Gewichten von Instanzen](#)
- [Konfigurieren einer Auto-Scaling-Gruppe zur Verwendung von Gewichtungen](#)
- [Beispiel: Spot-Preis pro Einheitsstunde](#)

Überlegungen

In diesem Abschnitt werden die wichtigsten Überlegungen zur effektiven Implementierung von Gewichtungen erörtert.

- Wählen Sie einige Instance-Typen aus, die den Leistungsanforderungen Ihrer Anwendung entsprechen. Entscheiden Sie anhand ihrer Fähigkeiten, welches Gewicht jeder Instance-Typ auf die gewünschte Kapazität Ihrer Auto Scaling Scaling-Gruppe angerechnet werden soll. Diese Gewichte gelten für aktuelle und future Fälle.
- Vermeiden Sie große Gewichtsunterschiede. Geben Sie beispielsweise nicht die Gewichtung 1 für einen Instance-Typ an, wenn der nächstgrößere Instance-Typ eine Gewichtung von 200 hat. Der Unterschied zwischen der kleinsten und der größten Gewichtung sollte auch nicht extrem sein. Extreme Gewichtsunterschiede können sich negativ auf die Optimierung von Kosten und Leistung auswirken.
- Geben Sie die gewünschte Kapazität der Gruppe in Einheiten und nicht in Instanzen an. Wenn Sie beispielsweise vCPU-basierte Gewichtungen verwenden, legen Sie die gewünschte Anzahl von Kernen sowie die Mindest- und Höchstzahl fest.
- Legen Sie die Gewichtungen und die gewünschte Kapazität so fest, dass die gewünschte Kapazität mindestens zwei- bis dreimal größer ist als Ihr größtes Gewicht.

Beachten Sie bei der Aktualisierung vorhandener Gruppen Folgendes:

- Wenn Sie einer vorhandenen Gruppe Gewichtungen hinzufügen, schließen Sie Gewichtungen für alle derzeit verwendeten Instance-Typen mit ein.

- Wenn Sie Gewichtungen hinzufügen oder ändern, startet oder beendet Amazon EC2 Auto Scaling Instances, um die gewünschte Kapazität auf der Grundlage der neuen Gewichtungswerte zu erreichen.
- Wenn Sie einen Instance-Typ entfernen, behalten laufende Instances dieses Typs ihre letzte Gewichtung, auch wenn sie nicht mehr definiert sind.

Verhalten beim Gewichten von Instanzen

Wenn Sie Instance-Gewichtungen verwenden, verhält sich Amazon EC2 Auto Scaling folgendermaßen:

- Die aktuelle Kapazität wird entweder bei der gewünschten Kapazität oder darüber liegen. Die aktuelle Kapazität kann die gewünschte Kapazität überschreiten, wenn Instances gestartet werden, die die verbleibenden gewünschten Kapazitätseinheiten überschreiten. Angenommen, Sie geben die zwei Instance-Typen `c5.2xlarge` und `c5.12xlarge` an und weisen für `c5.2xlarge` eine Instance-Gewichtung von „2“ und für `c5.12xlarge` eine von „12“ zu. Wenn 5 Einheiten übrig sind, um die gewünschte Kapazität zu erfüllen, und Amazon EC2 Auto Scaling eine `c5.12xlarge` bereitstellt, wird die gewünschte Kapazität um sieben Einheiten überschritten.
- Beim Starten von Instances priorisiert Amazon EC2 Auto Scaling die Verteilung der Kapazität auf die Availability Zones und die Einhaltung der Zuweisungsstrategien gegenüber der Überschreitung der gewünschten Kapazität.
- Amazon EC2 Auto Scaling kann die maximale Kapazitätsgrenze überschreiten, um das Gleichgewicht zwischen den Availability Zones aufrechtzuerhalten. Dabei werden Ihre bevorzugten Zuweisungsstrategien verwendet. Das von Amazon EC2 Auto Scaling erzwungene feste Limit ist Ihre gewünschte Kapazität zuzüglich Ihres größten Gewichts.

Konfigurieren einer Auto-Scaling-Gruppe zur Verwendung von Gewichtungen

Sie können eine Auto-Scaling-Gruppe für die Verwendung von Gewichtungen konfigurieren, wie in den folgenden AWS CLI Beispielen gezeigt. Weitere Informationen zur Verwendung der Konsole finden Sie unter [Erstellen Sie eine Gruppe mit gemischten Instances, indem Sie die Instance-Typen manuell auswählen](#).

So konfigurieren Sie eine Auto-Scaling-Gruppe zur Verwendung von Gewichtungen (AWS CLI)

Verwenden Sie den Befehl [create-auto-scaling-group](#). Der folgende Befehl erstellt zum Beispiel eine neue Auto-Scaling-Gruppe und weist Gewichtungen zu, indem er Folgendes angibt:

- Der Prozentsatz der Gruppe, die als On-Demand-Instances gestartet werden soll (0)
- Die Zuordnungsstrategie für Spot-Instances in jeder Availability Zone (capacity-optimized)
- Die in der Prioritätsreihenfolge zu startenden Instance-Typen (m4.16xlarge, m5.24xlarge)
- Die Instance-Gewichtungen, die dem relativen Größenunterschied (vCPUs) zwischen den Instance-Typen (16,) entsprechen 24
- Die Subnetze, in denen die Instances gestartet werden sollen (subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782), die jeweils einer anderen Availability Zone entsprechen
- Beschreibt eine Startvorlage (my-launch-template) und die Version der Startvorlage (\$Latest).

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Die Datei config.json enthält den folgenden Inhalt.

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
      "Overrides": [
        {
          "InstanceType": "m4.16xlarge",
          "WeightedCapacity": "16"
        },
        {
          "InstanceType": "m5.24xlarge",
          "WeightedCapacity": "24"
        }
      ]
    },
    "InstancesDistribution": {
      "OnDemandPercentageAboveBaseCapacity": 0,
      "SpotAllocationStrategy": "capacity-optimized"
    }
  },
  "MinSize": 160,
```

```
"MaxSize": 720,  
"DesiredCapacity": 480,  
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",  
"Tags": []  
}
```

So konfigurieren Sie eine vorhandene Auto-Scaling-Gruppe für die Verwendung von Gewichtungen (AWS CLI)

Verwenden Sie den Befehl [update-auto-scaling-group](#). Der folgende Befehl weist beispielsweise den Instance-Typen in einer bestehenden Auto-Scaling-Gruppe Gewichtungen zu, indem er Folgendes angibt:

- Die in der Prioritätsreihenfolge zu startenden Instance-Typen (c5.18xlarge, c5.24xlarge, c5.2xlarge, c5.4xlarge)
- Die Instance-Gewichtungen, die dem relativen Größenunterschied (vCPUs) zwischen den Instance-Typen (18,, 24,2,4) entsprechen
- Die neue, erhöhte gewünschte Kapazität, die größer als das größte Gewicht ist

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

Die Datei `config.json` enthält den folgenden Inhalt.

```
{  
  "AutoScalingGroupName": "my-existing-asg",  
  "MixedInstancesPolicy": {  
    "LaunchTemplate": {  
      "Overrides": [  
        {  
          "InstanceType": "c5.18xlarge",  
          "WeightedCapacity": "18"  
        },  
        {  
          "InstanceType": "c5.24xlarge",  
          "WeightedCapacity": "24"  
        },  
        {  
          "InstanceType": "c5.2xlarge",  
          "WeightedCapacity": "2"  
        }  
      ]  
    }  
  }  
}
```

```

        {
            "InstanceType": "c5.4xlarge",
            "WeightedCapacity": "4"
        }
    ]
}
},
"MinSize": 0,
"MaxSize": 100,
"DesiredCapacity": 100
}

```

So überprüfen Sie die Gewichtungen mithilfe der Befehlszeile

Verwenden Sie einen der folgenden Befehle:

- [describe-auto-scaling-groups](#) (AWS CLI)
- [Holen Sie sich- ASAuto ScalingGroup](#) (AWS Tools for Windows PowerShell)

Beispiel: Spot-Preis pro Einheitsstunde

Die folgende Tabelle vergleicht den stündlichen Preis für Spot-Instances in verschiedenen Availability Zones in USA Ost (Nord-Virginia) mit dem Preis für On-Demand-Instances in derselben Region. Bei den angezeigten Preisen handelt es sich um Beispielpreise und nicht um aktuelle Preise. Dies sind Ihre Kosten pro Instance-Stunde.

Beispiel: Spot-Preise pro Instance-Stunde

Instance-Typ	us-ost-1a	us-ost-1b	us-ost-1c	On-Demand-Preise
c5.2xlarge	0,180 US-Dollar	0,191 US-Dollar	0,170 US-Dollar	0,34 US-Dollar
c5.4xlarge	0,341 US-Dollar	0,361 US-Dollar	0,318 US-Dollar	0,68 US-Dollar
c5.12xlarge	0,779 US-Dollar	0,777 US-Dollar	0,777 US-Dollar	2,04 US-Dollar
c5.18xlarge	1,207 US-Dollar	1,475 US-Dollar	1,357 US-Dollar	3,06 US-Dollar

Instance-Typ	us-ost-1a	us-ost-1b	us-ost-1c	On-Demand-Preise
c5.24xlarge	1,555 US-Dollar	1,555 US-Dollar	1,555 US-Dollar	4,08 US-Dollar

Mit der Instance-Gewichtung können Sie Ihre Kosten auf Grundlage Ihrer Verwendung pro Einheitsstunde bewerten. Der Preis pro Einheitsstunde lässt sich bestimmen, indem der Preis für einen Instance-Typ durch die Anzahl an Einheiten geteilt wird, den er darstellt. Bei On-Demand-Instances entspricht der Preis pro Einheitsstunde bei der Bereitstellung eines Instance-Typs dem Preis der Bereitstellung desselben Instance-Typs einer anderen Größe. Im Gegensatz dazu variiert der Spot-Preis pro Einheitsstunde nach Spot-Pool.

Das folgende Beispiel zeigt, wie die Berechnung des Spot-Preises pro Stunde mit Instance-Gewichtungen funktioniert. Angenommen, Sie möchten Spot-Instances nur in us-east-1a starten. Der Preis pro Stunde wird in der folgenden Tabelle erfasst.

Beispiel: Spotpreis pro Stunde

Instance-Typ	us-ost-1a	Instance-Gewichtung	Preis pro Einheitsstunde
c5.2xlarge	0,180 US-Dollar	2	0,090 US-Dollar
c5.4xlarge	0,341 US-Dollar	4	0,085 US-Dollar
c5.12xlarge	0,779 US-Dollar	12	0,065 US-Dollar
c5.18xlarge	1,207 US-Dollar	18	0,067 US-Dollar
c5.24xlarge	1,555 US-Dollar	24	0,065 US-Dollar

Verwenden von mehreren Startvorlagen

Sie können nicht nur mehrere Instance-Typen verwenden, sondern auch mehrere Startvorlagen.

Nehmen wir an, Sie konfigurieren eine Auto-Scaling-Gruppe für rechenintensive Anwendungen und möchten eine Mischung aus C5-, C5a- und C6g-Instance-Typen einbeziehen. C6g-Instances verfügen jedoch über einen AWS Graviton-Prozessor, der auf der 64-Bit-Arm-Architektur basiert, während die C5- und C5a-Instances auf 64-Bit-Intel x86-Prozessoren laufen. Die Instances AMIs for

C5 und C5a funktionieren beide auf jeder dieser Instances, aber nicht auf C6g-Instances. Verwenden Sie eine andere Startvorlage für C6g-Instances, um dieses Problem zu lösen. Sie können immer noch dieselbe Startvorlage für C5- und C5a-Instances verwenden.

Dieser Abschnitt enthält Verfahren zur Verwendung von zur Ausführung von Aufgaben im Zusammenhang mit der AWS CLI Verwendung mehrerer Startvorlagen. Derzeit ist diese Funktion nur verfügbar, wenn Sie die AWS CLI oder ein SDK verwenden, und ist nicht von der Konsole aus verfügbar.

Inhalt

- [Konfigurieren einer Auto-Scaling-Gruppe zum Verwenden mehrerer Startvorlagen](#)
- [Zugehörige Ressourcen](#)

Konfigurieren einer Auto-Scaling-Gruppe zum Verwenden mehrerer Startvorlagen

Sie können eine Auto-Scaling-Gruppe so konfigurieren, dass sie mehrere Startvorlagen verwendet, wie in den folgenden Beispielen gezeigt.

So konfigurieren Sie eine neue Auto-Scaling-Gruppe für die Verwendung mehrerer Startvorlagen (AWS CLI)

Verwenden Sie den Befehl [create-auto-scaling-group](#). Mit dem folgenden Befehl wird zum Beispiel eine neue Auto-Scaling-Gruppe erstellt. Es gibt die Instance-Typen `c5.large`, `c5a.large` und `c6g.large` an und definiert eine neue Startvorlage für den Instance-Typ `c6g.large`, um sicherzustellen, dass ein geeignetes AMI zum Starten von Arm-Instances verwendet wird. Amazon EC2 Auto Scaling verwendet die Reihenfolge der Instance-Typen, um festzulegen, welcher Instance-Typ beim Erfüllen der On-Demand-Kapazität zuerst verwendet werden soll.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Die Datei `config.json` enthält den folgenden Inhalt.

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template-for-x86",
        "Version": "$Latest"
      }
    }
  }
}
```

```

    },
    "Overrides": [
      {
        "InstanceType": "c6g.large",
        "LaunchTemplateSpecification": {
          "LaunchTemplateName": "my-launch-template-for-arm",
          "Version": "$Latest"
        }
      },
      {
        "InstanceType": "c5.large"
      },
      {
        "InstanceType": "c5a.large"
      }
    ]
  },
  "InstancesDistribution": {
    "OnDemandBaseCapacity": 1,
    "OnDemandPercentageAboveBaseCapacity": 50,
    "SpotAllocationStrategy": "capacity-optimized"
  }
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
"Tags": [ ]
}

```

So konfigurieren Sie eine bestehende Auto-Scaling-Gruppe für die Verwendung mehrerer Startvorlagen (AWS CLI)

Verwenden Sie den Befehl [update-auto-scaling-group](#). Der folgende Befehl weist beispielsweise die Startvorlage namens *my-launch-template-for-arm* dem *c6g.large*-Instance-Typ für die Auto-Scaling-Gruppe namens *my-asg* zu.

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

Die Datei `config.json` enthält den folgenden Inhalt.

```
{
```

```

"AutoScalingGroupName":"my-asg",
"MixedInstancesPolicy":{
  "LaunchTemplate":{
    "Overrides":[
      {
        "InstanceType":"c6g.large",
        "LaunchTemplateSpecification": {
          "LaunchTemplateName": "my-launch-template-for-arm",
          "Version": "$Latest"
        }
      },
      {
        "InstanceType":"c5.large"
      },
      {
        "InstanceType":"c5a.large"
      }
    ]
  }
}

```

So konfigurieren Sie eine neue Auto Scaling Scaling-Gruppe für die Verwendung mehrerer Startvorlagen mit attributbasierter Instance-Typauswahl ()AWS CLI

Verwenden Sie den Befehl [create-auto-scaling-group](#). Der folgende Befehl erstellt beispielsweise eine neue Auto Scaling Scaling-Gruppe, indem er eine Startvorlage für AWS Graviton-Instances mit einem ARM-AMI und eine zusätzliche Startvorlage für AMD- oder Intel-basierte Instances mit einem x86-AMI angibt. Anschließend verwendet es zweimal die [attributbasierte Instance-Auswahl](#), um aus einer Vielzahl von Instance-Typen für jede CPU-Architektur auszuwählen. Mit dem [update-autoscaling-group](#) Befehl können Sie einer vorhandenen Auto Scaling Scaling-Gruppe eine ähnliche Konfiguration hinzufügen.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Die Datei `config.json` enthält den folgenden Inhalt.

```

{
  "AutoScalingGroupName":"my-asg",
  "MixedInstancesPolicy":{
    "LaunchTemplate":{

```

```

"LaunchTemplateSpecification":{
  "LaunchTemplateName":"my-launch-template-for-arm",
  "Version":"$Latest"
},
"Overrides":[
  {
    "InstanceRequirements": {
      "VCpuCount": {"Min": 2},
      "MemoryMiB": {"Min": 2048},
      "CpuManufacturers": ["amazon-web-services"]
    }
  },
  {
    "InstanceRequirements": {
      "VCpuCount": {"Min": 2},
      "MemoryMiB": {"Min": 2048},
      "CpuManufacturers": ["intel", "amd"]
    },
    "LaunchTemplateSpecification": {
      "LaunchTemplateName": "my-launch-template-for-x86",
      "Version": "$Latest"
    }
  }
]
},
"InstancesDistribution":{
  "OnDemandPercentageAboveBaseCapacity": 0,
  "SpotAllocationStrategy": "price-capacity-optimized"
},
"MinSize":1,
"MaxSize":10,
"DesiredCapacity":6,
"VPCZoneIdentifier":"subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
"Tags":[ ]
}

```

So überprüfen Sie die Startvorlagen für eine Auto-Scaling-Gruppe

Verwenden Sie einen der folgenden Befehle:

- [describe-auto-scaling-groups](#) (AWS CLI)
- [Holen Sie sich- ASAuto ScalingGroup](#) (AWS Tools for Windows PowerShell)

Zugehörige Ressourcen

[Ein Beispiel für die Angabe mehrerer Startvorlagen mithilfe der attributbasierten Instanztypauswahl finden Sie in einer CloudFormation Vorlage auf re:POST.AWS](#)

Erstellen Sie Auto-Scaling-Gruppen mit Startkonfigurationen

Important

Einschränkungen:

- Ab dem 1. Januar 2023 werden neue Amazon EC2 EC2-Instance-Typen in Startkonfigurationen nicht mehr unterstützt. Dies beinhaltet die Unterstützung für alle Instance-Typen, die AWS-Region nach dem Start der ersten Region hinzugefügt wurden.
- Konten, die am oder nach dem 1. Juni 2023 erstellt wurden, können keine neuen Startkonfigurationen über die Konsole erstellen.
- Konten, die am oder nach dem 1. Oktober 2024 erstellt wurden, können mit keiner Methode (Konsole AWS CLI, API oder CloudFormation) neue Startkonfigurationen erstellen.

Migrieren Sie zu Startvorlagen, um sicherzustellen, dass Sie weder jetzt noch in future neue Startkonfigurationen erstellen müssen. Informationen zum Migrieren Ihrer Auto-Scaling-Gruppen zu Startvorlagen finden Sie unter [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).

Wenn Sie eine Startkonfiguration oder eine EC2-Instance erstellt haben, können Sie eine Auto-Scaling-Gruppe erstellen, die eine Startkonfiguration als Konfigurationsvorlage für ihre EC2-Instances verwendet. Die Startkonfiguration gibt Informationen wie die AMI-ID, den Instance-Typ, das Schlüsselpaar, Sicherheitsgruppen und die Blockgerät-Zuweisung für Ihre Instances an. Weitere Informationen zum Erstellen von Startkonfigurationen finden Sie unter [Erstellen einer Startkonfiguration](#).

Sie müssen über IAM-Berechtigungen verfügen, um eine Auto-Scaling-Gruppe zu erstellen. Sie müssen auch über ausreichende Berechtigungen verfügen, um die serviceverknüpfte Rolle zu erstellen, die Amazon EC2 Auto Scaling verwendet, um Aktionen in Ihrem Namen durchzuführen, falls sie noch nicht existiert. Beispiele für IAM-Richtlinien, die ein Administrator als Referenz für

die Erteilung von Berechtigungen verwenden kann, finden Sie unter [Beispiele für identitätsbasierte Richtlinien](#).

Inhalt

- [Eine Auto-Scaling-Gruppe mithilfe einer Startkonfiguration erstellen](#)
- [Erstellen Sie eine Auto Scaling Scaling-Gruppe aus einer vorhandenen Instanz mit dem AWS CLI](#)

Eine Auto-Scaling-Gruppe mithilfe einer Startkonfiguration erstellen

Important

Wir stellen Informationen zu Startkonfigurationen für Kunden bereit, die noch nicht von Startkonfigurationen zu Startvorlagen migriert sind. Informationen zum Migrieren Ihrer Auto-Scaling-Gruppen zu Startvorlagen finden Sie unter [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).

Wenn Sie eine Auto-Scaling-Gruppe erstellen, müssen Sie die notwendigen Informationen zur Konfiguration der Amazon EC2-Instances, die Availability Zones und VPC-Subnetze für die Instances, die gewünschte Kapazität sowie die minimalen und maximalen Kapazitätsgrenzen angeben.

Das folgende Verfahren veranschaulicht das Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startkonfiguration. Sie können eine Startkonfiguration nicht ändern, nachdem sie erstellt wurde. Sie können jedoch die Startkonfiguration für eine Auto-Scaling-Gruppe ersetzen. Weitere Informationen finden Sie unter [Ändern der Startkonfiguration für eine Auto-Scaling-Gruppe](#).

Voraussetzungen

- Sie müssen eine Startkonfiguration erstellt haben. Weitere Informationen finden Sie unter [Erstellen einer Startkonfiguration](#).

Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startkonfiguration (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.

2. Wählen Sie in der Navigationsleiste oben auf dem Bildschirm dieselbe aus, AWS-Region die Sie bei der Erstellung der Startkonfiguration verwendet haben.
3. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
4. Geben Sie auf der Seite Startvorlage oder -konfiguration auswählen für Auto-Scaling-Gruppenname einen Namen für Ihre Auto-Scaling-Gruppe ein.
5. Gehen Sie folgendermaßen vor, um eine Startkonfiguration auszuwählen:
 - a. Wählen Sie unter Launch Template (Startvorlage) die Option Switch to launch configuration (Wechsel zur Startkonfiguration) aus.
 - b. Wählen Sie unter Launch configuration (Startkonfiguration) eine vorhandene Startkonfiguration aus.
 - c. Stellen Sie sicher, dass Ihre Startkonfiguration alle Optionen unterstützt, die Sie verwenden möchten, und wählen Sie dann Next (Weiter) aus.
6. Wählen Sie auf der Seite Instance-Startoptionen konfigurieren unter Netzwerk für VPC eine VPC aus. Die Auto-Scaling-Gruppe muss in derselben VPC erstellt werden wie die Sicherheitsgruppe, die Sie in Ihrer Startkonfiguration angegeben haben.
7. Für Availability Zones und Subnets (Subnetze) wählen Sie ein oder mehrere Subnetze in der angegebenen VPC aus. Verwenden Sie Subnetze in mehreren Availability Zones, um eine hohe Verfügbarkeit zu erzielen. Weitere Informationen finden Sie unter [Überlegungen bei der Auswahl von VPC-Subnetzen](#).
8. Wählen Sie Weiter aus.

Oder akzeptieren Sie die weiteren Standardwerte, und klicken Sie dann auf Skip to review (Mit Prüfen fortfahren).
9. (Optional) Konfigurieren Sie auf der Seite Konfigurieren von erweiterten Optionen die folgenden Optionen und wählen Sie Weiter:
 - a. (Optional) Wählen Sie für Integritätsprüfungen und Zusätzliche Zustandsprüfungsarten die Option Amazon EBS-Zustandsprüfungen aktivieren aus. Weitere Informationen finden Sie unter [Überwachen Sie Auto Scaling Scaling-Instances mit beeinträchtigten Amazon EBS-Volumes mithilfe von Zustandsprüfungen](#).
 - b. (Optional) Geben Sie unter Karenzzeit für die Zustandsprüfung die Zeit in Sekunden ein. Diese Zeitspanne gibt an, wie lange Amazon EC2 Auto Scaling warten muss, bevor der Integritätsstatus einer Instance überprüft wird, nachdem sie den InService Status erreicht

- hat. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).
- c. Wählen Sie unter Zusätzliche Einstellungen, Überwachung, aus, ob die Erfassung von CloudWatch Gruppenmetriken aktiviert werden soll. Diese Metriken liefern Messwerte, die Indikatoren für ein potenzielles Problem sein können, wie z.B. die Anzahl der abgebrochenen Instances oder die Anzahl der ausstehenden Instances. Weitere Informationen finden Sie unter [Überwachen Sie CloudWatch Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances](#).
 - d. Wählen Sie unter Standardinstanzaufwärmen aktivieren diese Option und wählen Sie die Aufwärmzeit für Ihre Anwendung aus. Wenn Sie eine Auto Scaling-Gruppe mit einer Skalierungsrichtlinie erstellen, verbessert die Standard-Instance-Aufwärmfunktion die CloudWatch Amazon-Metriken, die für die dynamische Skalierung verwendet werden. Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).
10. Konfigurieren Sie auf der Seite Configure group size and scaling policies (Gruppengröße und Skalierungsrichtlinien konfigurieren) die folgenden Optionen, und wählen Sie dann Next (Weiter):
- a. Geben Sie unter Gruppengröße für Gewünschte Kapazität die anfängliche Anzahl von Instances ein, die gestartet werden sollen.
 - b. Wenn im Abschnitt Skalierung unter Skalierungslimits Ihr neuer Wert für die gewünschte Kapazität größer als die gewünschte Mindestkapazität und die gewünschte Höchstkapazität ist, wird die gewünschte Höchstkapazität automatisch auf den neuen Wert für die gewünschte Kapazität erhöht. Sie können die Limits bei Bedarf ändern. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).
 - c. Wählen Sie für Automatische Skalierung aus, ob Sie eine Skalierungsrichtlinie für die Zielverfolgung erstellen möchten. Sie können diese Richtlinie auch erstellen, nachdem Sie Ihre Auto-Scaling-Gruppe erstellt haben.

Wenn Sie sich für die Skalierungsrichtlinie für die Zielverfolgung entscheiden, befolgen Sie die Anweisungen unter [Erstellen einer Zielverfolgungs-Skalierungsrichtlinie](#), um die Richtlinie zu erstellen.

- d. Wählen Sie unter Instance-Wartungsrichtlinie aus, ob Sie eine Instance-Wartungsrichtlinie erstellen möchten. Sie können diese Richtlinie auch erstellen, nachdem Sie Ihre Auto-Scaling-Gruppe erstellt haben. Befolgen Sie zum Erstellen der Richtlinie die Anweisungen unter [Festlegen einer Instance-Wartungsrichtlinie](#).

- e. Wählen Sie unter Instance scale-in protection (Instance-Skalierungsschutz), ob der Instance-Skalierungsschutz aktiviert werden soll. Weitere Informationen finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).
11. (Optional) Um Benachrichtigungen zu erhalten, konfigurieren Sie für Add notification (Benachrichtigungen hinzufügen) die Benachrichtigung und wählen Sie anschließend Next (Weiter) aus. Weitere Informationen finden Sie unter [Amazon SNS SNS-Benachrichtigungsoptionen für Amazon EC2 Auto Scaling](#).
12. (Optional) Um Tags hinzuzufügen, wählen Sie Add tag (Tag hinzufügen) aus, geben Sie für jedes Tag einen Tag-Schlüssel und einen Wert an und wählen Sie anschließend Next (Weiter) aus. Weitere Informationen finden Sie unter [Tagging von Auto-Scaling-Gruppen und Instances](#).
13. Wählen Sie auf der Seite Review (Prüfen) Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Erstellen Sie wie folgt eine Auto-Scaling-Gruppe über die Befehlszeile:

Verwenden Sie einen der folgenden Befehle:

- [create-auto-scaling-group](#) (AWS CLI)
- [Neu- ASAuto ScalingGroup](#) (AWS Tools for Windows PowerShell)

Erstellen Sie eine Auto Scaling Scaling-Gruppe aus einer vorhandenen Instanz mit dem AWS CLI

Important

Wir stellen Informationen zu Startkonfigurationen für Kunden bereit, die noch nicht von Startkonfigurationen zu Startvorlagen migriert sind. Informationen zum Migrieren Ihrer Auto-Scaling-Gruppen zu Startvorlagen finden Sie unter [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).

Wenn Sie zum ersten Mal eine Auto-Scaling-Gruppe erstellen, empfehlen wir Ihnen, die Konsole zu verwenden, um eine Startvorlage aus einer bestehenden EC2-Instanz zu erstellen. Verwenden Sie dann die Startvorlage, um eine neue Auto-Scaling-Gruppe zu erstellen. Informationen zu

diesen Verfahren finden Sie unter [Erstellen einer Auto-Scaling-Gruppe mithilfe des Amazon EC2-Startassistenten](#).

Das folgende Verfahren zeigt, wie Sie eine Auto-Scaling-Gruppe erstellen, indem Sie eine vorhandene Instance angeben, die als Basis zum Starten anderer Instances verwendet werden soll. Für die Erstellung einer EC2-Instance sind mehrere Parameter erforderlich, z. B. die Amazon Machine Image (AMI) ID, der Instanztyp, das Schlüsselpaar und die Sicherheitsgruppe. Alle diese Informationen werden auch von Amazon EC2 Auto Scaling benutzt, um Instances in Ihrem Namen zu starten, wenn eine Skalierung erforderlich ist. Diese Informationen werden entweder in einer Startvorlage oder in einer Startkonfiguration gespeichert.

Wenn Sie eine vorhandene Instance verwenden, erstellt Amazon EC2 Auto Scaling eine Auto-Scaling-Gruppe, die Instances auf der Grundlage einer gleichzeitig erstellten Startkonfiguration startet. Die neue Startkonfiguration hat denselben Namen wie die Auto-Scaling-Gruppe und enthält bestimmte Konfigurationsdetails der identifizierten Instance.

Die folgenden Konfigurationsdetails werden von der identifizierten Instance in die Startkonfiguration kopiert:

- AMI-ID
- Instance-Typ
- Schlüsselpaar
- Sicherheitsgruppen
- Typ der IP-Adresse (öffentlich oder privat)
- IAM-Instance-Profil, falls zutreffend
- Überwachung (richtig oder falsch)
- EBS optimiert (richtig oder falsch)
- Tenancy-Einstellung beim Start in einer VPC (geteilt oder dediziert)
- Kernel-ID und RAM-Datenträger-ID, falls zutreffend
- Benutzerdaten, falls angegeben
- Spotpreis (maximal)

Das VPC-Subnetz und die Availability Zone werden von der identifizierten Instance in die eigene Ressourcendefinition der Auto-Scaling-Gruppe kopiert.

Wenn sich die identifizierte Instance in einer Platzierungsgruppe befindet, startet die neue Auto-Scaling-Gruppe Instanzen in dieselbe Platzierungsgruppe wie die identifizierte Instance. Da die Startkonfigurationseinstellungen die Angabe einer Platzierungsgruppe nicht zulassen, wird die Platzierungsgruppe in das `PlacementGroup`-Attribut der neuen Auto-Scaling-Gruppe kopiert.

Die folgenden Konfigurationsdetails werden nicht von Ihrer identifizierten Instance übernommen:

- Speicher: Die Blockgeräte (EBS-Volumen und Instance-Speichervolumen) werden nicht von der identifizierten Instance kopiert. Stattdessen bestimmt die bei der Erstellung des AMI erstellte Blockgerät-Zuweisung, welche Geräte verwendet werden.
- Anzahl der Netzwerkschnittstellen: Die Netzwerkschnittstellen werden nicht von Ihrer identifizierten Instance kopiert. Stattdessen verwendet Amazon EC2 Auto Scaling die Standardeinstellungen, um eine Netzwerkschnittstelle zu erstellen, nämlich die primäre Netzwerkschnittstelle (`eth0`).
- Optionen für Instance-Metadaten: Die Einstellungen für die zugänglichen Metadaten, die Metadatenversion und das Sprunglimit für Token-Antworten werden nicht von der identifizierten Instance übernommen. Stattdessen verwendet Amazon EC2 Auto Scaling seine Standardeinstellungen. Weitere Informationen finden Sie unter [Konfigurieren der Instance-Metadaten-Optionen](#).
- Lastenverteilung: Ist die identifizierte Instance bei mindestens einem Load Balancer angemeldet, werden die Informationen über den Load Balancer nicht automatisch in den Load Balancer oder in das Zielgruppenattribut der neuen Auto-Scaling-Gruppe kopiert.
- Tags: Verfügt die identifizierte Instance über Tags, werden diese nicht in das Attribut `Tags` der neuen Auto-Scaling-Gruppe kopiert.

Voraussetzungen

Die EC2-Instance muss die folgenden Kriterien erfüllen:

- Die Instance gehört keiner anderen Auto-Scaling-Gruppe an.
- Der Status der Instance lautet `running`.
- Das AMI zum Starten der Instance muss noch vorhanden sein.

Erstellen einer Auto-Scaling-Gruppe aus einer EC2-Instance (AWS CLI)

Das folgende Verfahren demonstriert die Verwendung eines CLI-Befehls zum Erstellen einer Auto-Scaling-Gruppe aus einer EC2-Instance.

Bei diesem Verfahren wird die Instance nicht zur Auto-Scaling-Gruppe hinzugefügt. Damit die Instance angefügt werden kann, müssen Sie den Befehl [attach-instances](#) ausführen, nachdem die Auto-Scaling-Gruppe erstellt wurde.

Bevor Sie beginnen, suchen Sie die ID der EC2-Instance mithilfe der Amazon-EC2-Konsole oder dem Befehl [describe-instances](#).

So verwenden Sie die aktuelle Instance als Vorlage

- Verwenden Sie den folgenden [create-auto-scaling-group](#) Befehl, um eine Auto Scaling Scaling-Gruppe, `my-asg-from-instance`, aus der EC2-Instance `i-123456789abcdefg0` zu erstellen.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-from-instance \  
  --instance-id i-123456789abcdefg0 --min-size 1 --max-size 2 --desired-capacity 2
```

So prüfen Sie, dass Ihre Auto-Scaling-Gruppe eine neue Instance gestartet hat

- Verwenden Sie den folgenden [describe-auto-scaling-groups](#) Befehl, um zu überprüfen, ob die Auto Scaling Scaling-Gruppe erfolgreich erstellt wurde.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg-from-instance
```

Die folgende Beispielantwort zeigt, dass die gewünschte Kapazität der Gruppe 2 beträgt, die Gruppe über zwei laufende Instances verfügt und die Startkonfiguration den Namen `my-asg-from-instance` hat.

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "my-asg-from-instance",  
      "AutoScalingGroupARN": "arn",  
      "LaunchConfigurationName": "my-asg-from-instance",  
      "MinSize": 1,  
      "MaxSize": 2,  
      "DesiredCapacity": 2,  
      "DefaultCooldown": 300,  
      "AvailabilityZones": [  

```

```
    "us-west-2a"
  ],
  "LoadBalancerNames": [],
  "TargetGroupARNs": [],
  "HealthCheckType": "EC2",
  "HealthCheckGracePeriod": 0,
  "Instances": [
    {
      "InstanceId": "i-34567890abcdef012",
      "InstanceType": "t2.micro",
      "AvailabilityZone": "us-west-2a",
      "LifecycleState": "InService",
      "HealthStatus": "Healthy",
      "LaunchConfigurationName": "my-asg-from-instance",
      "ProtectedFromScaleIn": false
    },
    {
      "InstanceId": "i-012345abcdefg6789",
      "InstanceType": "t2.micro",
      "AvailabilityZone": "us-west-2a",
      "LifecycleState": "InService",
      "HealthStatus": "Healthy",
      "LaunchConfigurationName": "my-asg-from-instance",
      "ProtectedFromScaleIn": false
    }
  ],
  "CreatedTime": "2020-10-28T02:39:22.152Z",
  "SuspendedProcesses": [ ],
  "VPCZoneIdentifier": "subnet-0abc1234",
  "EnabledMetrics": [ ],
  "Tags": [ ],
  "TerminationPolicies": [
    "Default"
  ],
  "NewInstancesProtectedFromScaleIn": false,
  "ServiceLinkedRoleARN": "arn",
  "TrafficSources": [ ]
}
]
```

So zeigen Sie die Startkonfiguration an

- Verwenden Sie den folgenden [describe-launch-configurations](#) Befehl, um die Details der Startkonfiguration anzuzeigen.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-asg-from-instance
```

Das Folgende ist eine Beispielausgabe:

```
{
  "LaunchConfigurations": [
    {
      "LaunchConfigurationName": "my-asg-from-instance",
      "LaunchConfigurationARN": "arn",
      "ImageId": "ami-234567890abcdefgh",
      "KeyName": "my-key-pair-uswest2",
      "SecurityGroups": [
        "sg-12abcdefgh3456789"
      ],
      "ClassicLinkVPCSecurityGroups": [ ],
      "UserData": "",
      "InstanceType": "t2.micro",
      "KernelId": "",
      "RamdiskId": "",
      "BlockDeviceMappings": [ ],
      "InstanceMonitoring": {
        "Enabled": true
      },
      "CreatedTime": "2020-10-28T02:39:22.321Z",
      "EbsOptimized": false,
      "AssociatePublicIpAddress": true
    }
  ]
}
```

Beenden der Instances

- Sie können die Instance beenden, wenn Sie sie nicht mehr benötigen. Der folgende [terminate-instances](#)-Befehl beendet die Instance `i-123456789abcdefgh0`.

```
aws ec2 terminate-instances --instance-ids i-123456789abcdefg0
```

Wenn Sie eine Amazon EC2-Instance beenden, kann diese nicht neu gestartet werden. Nach dem Beenden sind die Daten nicht mehr vorhanden und das Volume kann nicht an eine Instance angefügt werden. Weitere Informationen zum Beenden von Instances finden Sie unter [Eine Instance beenden](#) im Amazon EC2 EC2-Benutzerhandbuch.

Starten Sie Instances synchron

Amazon EC2 Auto Scaling bietet zwei Methoden zum Starten von Instances in Ihrer Auto Scaling Scaling-Gruppe: asynchrones Skalierungsverhalten und synchrone Bereitstellung mithilfe der API. `LaunchInstances`

Bei der synchronen Bereitstellung verwenden Sie die `LaunchInstances` API, um eine bestimmte Anzahl von Instances in einer bestimmten Availability Zone anzufordern. Die synchrone Bereitstellung bietet die folgenden Vorteile:

- Sofortiges Feedback zur Kapazitätsverfügbarkeit in bestimmten Availability Zones
- Präzise Kontrolle darüber, in welcher Availability Zone Instances gestartet werden
- Deterministische Instanz IDs zur sofortigen Verwendung in Orchestrierungssystemen
- Skalierungsentscheidungen in Echtzeit auf der Grundlage von tatsächlichen Kapazitätsbeschränkungen
- Schnellere Skalierung durch Eliminierung von Wartezeiten für asynchrone Auto Scaling Scaling-Starts

Wenn Sie bei asynchronem Auto Scaling die gewünschte Kapazität ändern oder wenn eine Skalierungsrichtlinie ausgelöst wird, verarbeitet Amazon EC2 Auto Scaling die Skalierungsanforderung und startet Instances im Hintergrund. Sie müssen die Skalierungsaktivitäten überwachen oder Ihre Auto Scaling Scaling-Gruppe beschreiben, um festzustellen, wann Instances erfolgreich gestartet wurden.

Note

- Die `LaunchInstances` API funktioniert nur mit Auto Scaling Scaling-Gruppen, die Startvorlagen verwenden. Auto Scaling Scaling-Gruppen, die Startkonfigurationen

verwenden, werden nicht unterstützt. Wenn Ihre Auto Scaling Scaling-Gruppe eine Startkonfiguration verwendet, müssen Sie zu einer Startvorlage migrieren, bevor Sie die synchrone Bereitstellung verwenden können.

- Die LaunchInstances API unterstützt Richtlinien für gemischte Instanzen mit ausschließlich On-Demand-Kaufoptionen oder vollständigen Spot-Kaufoptionen. Gemischte Richtlinien, die sowohl On-Demand-Instances als auch Spot-Instances kombinieren, werden nicht unterstützt.
- Für Auto Scaling Scaling-Gruppen, die mehrere Availability Zones abdecken, müssen Sie die Availability Zone oder das Zielsubnetz angeben. Für Single-AZ-Gruppen ist dieser Parameter optional.

Synchrone Bereitstellung und asynchrone Skalierung

Synchrone Bereitstellung

Wenn Sie die LaunchInstances API verwenden, geht Amazon EC2 Auto Scaling wie folgt vor:

- Versucht sofort, die angeforderten Instances zu starten mit CreateFleet
- Wartet, CreateFleet bis die Instanz zurückgegeben wird, IDs bevor sie reagiert
- Gibt bei Erfolg Informationen zu Instance IDs, Instance-Typen und Availability Zone zurück
- Gibt spezifische Fehlercodes und Details zum Fehler zurück
- Bietet sofortiges Feedback und ermöglicht Skalierungsentscheidungen in Echtzeit

Asynchrone Skalierung

Wenn Sie asynchrone Auto Scaling-Methoden verwenden, z. B. die gewünschte Kapazität ändern oder Skalierungsrichtlinien verwenden, kann Amazon EC2 Auto Scaling:

- Aktualisiert die gewünschte Kapazität in der API, gibt aber nicht sofort Instances zurück
- Plant, dass die Instanz automatisch in allen Availability Zones startet
- Startet Instanzen über Hintergrund-Workflows
- Verteilt die Kapazität automatisch auf mehrere Availability Zones und sorgt so für ein ausgewogenes Verhältnis
- Behandelt Startfehler mit integrierter Wiederholungslogik

Sie müssen Skalierungsaktivitäten abfragen oder Ihre Auto Scaling Scaling-Gruppe beschreiben, um den Status der Startvorgänge zu überprüfen.

Einschränkungen und Überlegungen

Beachten Sie bei der Arbeit mit synchroner Bereitstellung die folgenden Hinweise und Einschränkungen:

- **Instanzstatus nach dem Start** — Von der API zurückgegebene Instanzen befinden sich im Status „Ausstehend“. Sie können bei nachfolgenden Workflow-Prozessen oder Lifecycle-Hooks immer noch fehlschlagen. Eine erfolgreiche API-Antwort bedeutet, dass EC2 die Startanforderung akzeptiert und die Instance-IDs zurückgegeben hat. Instances werden nicht automatisch als vollständig bereit für Workloads betrachtet und müssen die standardmäßigen EC2- und Auto Scaling Scaling-Lebenszyklusprozesse abschließen.
- **Beschränkung auf warme Pools** — Auto Scaling Scaling-Gruppen mit warmen Pools werden derzeit nicht unterstützt. Wenn Sie versuchen, die LaunchInstances API in einer Auto Scaling Scaling-Gruppe aufzurufen, für die ein warmer Pool konfiguriert ist, führt die API einen Kaltstart durch, anstatt Warm-Pool-Instances zu verwenden, und gibt einen `UnsupportedOperation` Fehler zurück. Weitere Informationen zu Kaltstarts finden Sie unter [Einschränkungen von Warmpools](#).
- **API-Timeout und Wiederholungsversuche** — Wenn der zugrunde liegende CreateFleet Vorgang länger als erwartet dauert, kann es sein, dass die API eine Zeitüberschreitung durchführt und ein Idempotenz-Token zurückgibt. Sie können es erneut versuchen, indem Sie dasselbe verwenden ClientToken , um den ursprünglichen Startvorgang nachzuverfolgen, oder describe-instances mit dem Client-Token verwenden, um gestartete Instances zu überprüfen.
- **Einschränkungen der Availability Zone** — Wenn sich Ihre Auto Scaling Scaling-Gruppe über mehrere Availability Zones erstreckt und Availability Zone Rebalancing aktiviert ist, kann das synchrone Starten von Instances zu Betriebskonflikten führen:
 - **Einzelne AZ-Beschränkung pro Anruf** — Jeder LaunchInstances API-Aufruf kann nur auf eine Availability Zone abzielen, auch wenn sich Ihre Auto Scaling Scaling-Gruppe über mehrere Zonen erstreckt.
 - **Konflikte beim AZ-Rebalancing** — Wenn in Ihrer Auto Scaling Scaling-Gruppe das AZ-Rebalancing aktiviert ist, AZs können sequentielle Aufrufe zwischen verschiedenen Geräten zusätzliche asynchrone Starts auslösen, was zu mehr Instances als beabsichtigt führt. Erwägen Sie, das AZ-Rebalancing auszusetzen, um die Kapazität präzise zu steuern. Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).

- **Teilweise Erfolgsszenarien** — Die `LaunchInstances` API gibt möglicherweise einen teilweisen Erfolg zurück, wenn nur ein Teil der angeforderten Kapazität verfügbar ist, was dem normalen EC2-Verhalten entspricht. Die API gibt erfolgreich gestartete Instances zusammen mit Fehlerdetails für fehlgeschlagene Starts zurück. In Anwendungsfällen, bei denen alle Instances zusammen gestartet werden müssen (z. B. Anwendungen, die für eine geringe Latenz alle Instances in derselben AZ benötigen), müssen Sie teilweise gestartete Instances beenden und es in einer anderen AZ erneut versuchen. Berücksichtigen Sie dieses Verhalten, wenn Sie die Wiederholungslogik für kapazitätssensitive Workloads entwerfen.
- **Instanzgewichte** — Wenn Ihre Auto Scaling Scaling-Gruppe Instance-Gewichtungen verwendet, steht der `RequestedCapacity` Parameter für gewichtete Kapazitätseinheiten, nicht für die Anzahl der Instances. Die tatsächliche Anzahl der gestarteten Instances hängt von den ausgewählten Instance-Typen und ihren konfigurierten Gewichtungen ab. EC2 Auto Scaling begrenzt Starts auf 100 Instances pro API-Aufruf, unabhängig von der angeforderten gewichteten Kapazität.
- **Gemischte Instanztypen** — Die `LaunchInstances` API verwendet die bestehende Richtlinie für gemischte Instanzen Ihrer Auto Scaling Scaling-Gruppe, um zu bestimmen, welche Instance-Typen gestartet werden sollen. Die API startet Instances entsprechend der Zuweisungsstrategie und den Prioritäten Ihrer Gruppe für den Instanztyp.

Instances mit synchroner Bereitstellung werden gestartet

Sie können die `LaunchInstances` API verwenden, um synchron eine bestimmte Anzahl von Instances in Ihrer Auto Scaling Scaling-Gruppe zu starten. Die API startet Instances in der von Ihnen angegebenen Availability Zone oder dem Subnetz und gibt sofort die Instance-IDs oder Fehlerinformationen zurück.

Voraussetzungen

Bevor Sie die `LaunchInstances` API verwenden können, müssen Sie über Folgendes verfügen:

- Eine Auto Scaling Scaling-Gruppe, die eine Startvorlage verwendet (Startkonfigurationen werden nicht unterstützt)
- Sie benötigen Berechtigungen für die folgenden IAM-Aktionen:
 - `autoscaling:LaunchInstances`
 - `ec2:CreateFleet`
 - `ec2:DescribeLaunchTemplateVersions`

Starten Sie Instances mit synchroner Bereitstellung

Sie können Instances mit synchroner Bereitstellung über den `start`-Befehl der AWS CLI

AWS CLI

So starten Sie Instances mit synchroner Bereitstellung:

```
aws autoscaling launch-instances \  
  --auto-scaling-group-name group-name \  
  --requested-capacity number \  
  [--availability-zones zone-name] \  
  [--subnet-ids subnet-id] \  
  [--availability-zone-ids zone-id] \  
  [--retry-strategy none|retry-with-group-configuration] \  
  [--client-token token]
```

Beispiele

Instanzen in einer bestimmten Availability Zone starten

```
aws autoscaling launch-instances \  
  --auto-scaling-group-name my-asg \  
  --requested-capacity 3 \  
  --availability-zones us-east-1a \  
  --retry-strategy retry-with-group-configuration
```

Instances in einem bestimmten Subnetz starten

```
aws autoscaling launch-instances \  
  --auto-scaling-group-name my-asg \  
  --requested-capacity 2 \  
  --subnet-ids subnet-12345678 \  
  --retry-strategy none \  
  --client-token my-unique-token-123
```

Umgang mit Antworten

Beispiel für eine erfolgreiche Antwort:

```
{
```

```

"AutoScalingGroupName": "my-asg",
"ClientToken": "my-unique-token-123",
"Instances": [
  {
    "InstanceType": "m5.xlarge",
    "AvailabilityZone": "us-east-1a",
    "AvailabilityZoneId": "use1-az1",
    "SubnetId": "subnet-12345678",
    "MarketType": "OnDemand",
    "InstanceIds": ["i-0123456789abcdef0", "i-0fedcba9876543210"]
  }
],
"Errors": []
}

```

Beispiel für eine Antwort mit Fehlern

```

{
  "AutoScalingGroupName": "my-asg",
  "ClientToken": "my-unique-token-123",
  "Instances": [],
  "Errors": [
    {
      "InstanceType": "m5.large",
      "AvailabilityZone": "us-east-1a",
      "AvailabilityZoneId": "use1-az1",
      "SubnetId": "subnet-12345678",
      "MarketType": "OnDemand",
      "ErrorCode": "InsufficientInstanceCapacity",
      "ErrorMessage": "There is not enough capacity to fulfill your request for instance type 'm5.large' in 'us-east-1a'"
    }
  ]
}

```

Behandeln Sie Startfehler und Wiederholungsversuche

Wenn die `LaunchInstances` API auf Fehler stößt, können Sie Wiederholungsstrategien mithilfe von Idempotenz-Tokens und entsprechenden Wiederholungsrichtlinien implementieren.

Sie können den Parameter `client-token` verwenden, um Anfragen erneut zu versuchen. Sie können auch die folgenden Wiederholungsstrategien verwenden:

- `RetryStrategy: none`(Standard) — Wenn der API-Aufruf fehlschlägt, bleibt die gewünschte Kapazität der Auto Scaling Scaling-Gruppe unverändert und es erfolgt kein automatischer Wiederholungsversuch.
- `RetryStrategy: retry-with-group-configuration` - Wenn der API-Aufruf fehlschlägt, wird die gewünschte Kapazität der Auto Scaling-Gruppe um den angeforderten Betrag erhöht, und Auto Scaling versucht automatisch erneut, Instances mit der Standardkonfiguration und den Standardprozessen der Gruppe zu starten.

Das Wiederholungsverhalten für `RetryStrategy: retry-with-group-configuration` hängt von der Art des Fehlers ab:

- **Validierungsfehler:** Die gewünschte Kapazität wurde nicht erhöht, da der Vorgang nicht fortgesetzt werden kann. Zum Beispiel ungültige Parameter oder nicht unterstützte Konfigurationen.
- **Kapazitätsfehler:** Die gewünschte Kapazität wird erhöht und Auto Scaling versucht erneut, Instances asynchron mit den normalen Skalierungsprozessen der Gruppe zu starten.

Verwendung von Client-Token zur Bekämpfung von Idempotenz

Der `client-token` Parameter gewährleistet idempotente Operationen und ermöglicht sichere Wiederholungen von Startanfragen.

Wichtigste Verhaltensweisen:

- Client-Token haben eine Gültigkeitsdauer von 8 Stunden ab der ersten Anfrage
- Bei einem erneuten Versuch mit demselben Client-Token innerhalb von 8 Stunden wird die zwischengespeicherte Antwort zurückgegeben, anstatt neue Instances zu starten
- Nach 8 Stunden initiiert dasselbe Client-Token einen neuen Startvorgang

Aktualisieren einer Auto-Scaling-Gruppe

Sie können die meisten Details Ihrer Auto-Scaling-Gruppe aktualisieren. Sie können den Namen einer Auto Scaling Scaling-Gruppe nicht aktualisieren oder ändern AWS-Region.

So aktualisieren Sie eine Auto-Scaling-Gruppe (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Ihre Auto-Scaling-Gruppe aus, um Informationen über die Gruppe anzuzeigen, mit Registerkarten für Details, Aktivität, Automatische Skalierung, Instance-Verwaltung, Überwachung und Instance-Aktualisierung.
3. Wählen Sie die Registerkarten für die gewünschten Konfigurationsbereiche aus und aktualisieren Sie die Einstellungen nach Bedarf. Wählen Sie für jede Einstellung, die Sie bearbeiten, Aktualisieren aus, um Ihre Änderungen an der Konfiguration der Auto-Scaling-Gruppe zu speichern.

- Registerkarte Details

Dies sind die allgemeinen Einstellungen für Ihre Auto-Scaling-Gruppe. Sie können diese auf dieselbe Weise bearbeiten und verwalten wie bei der Erstellung der Auto-Scaling-Gruppe.

Der Abschnitt Erweiterte Konfigurationen enthält einige Optionen, die bei der Erstellung der Gruppe nicht verfügbar sind, z. B. [Beendigungsrichtlinien](#), [Abkühlungsphase](#), [Ausgesetzte Prozesse](#) und [Maximale Instance-Lebensdauer](#). Sie können auch die Platzierungsgruppe und die [serviceverknüpfte Rolle](#) der Auto-Scaling-Gruppe anzeigen, aber nicht bearbeiten.

- Registerkarte „Integrationen“

- Lastenausgleich — [Elastic Load Balancing](#)

Wenn die Gruppe mit Elastic Load Balancing-Ressourcen verbunden ist, lesen Sie vor dem Ändern der Availability Zones bitte [Fügen Sie eine Availability Zone hinzu](#). Einige Einschränkungen auf dem Load Balancer können Sie daran hindern, Änderungen an den Availability Zones Ihrer Gruppe auf die Availability Zones Ihres Load Balancers anzuwenden.

- Optionen für die Integration von VPC Lattice — [VPC Lattice](#)

- ARC-Zonenverschiebung — [Auto Scaling Scaling-Gruppenzonenverschiebung](#)

- Registerkarte Automatische Skalierung

- Dynamische Skalierungsrichtlinien — [Dynamische](#) Skalierungsrichtlinien

- Richtlinien für vorausschauende Skalierung — Richtlinien für [prädiktive](#) Skalierung

- Geplante Aktionen — [Geplante Aktionen](#)

- Registerkarte Instance-Verwaltung

- ~~Lebenszyklus Hooks~~ — [Lebenszyklus Hooks](#)

- Warmer Pool — [Warme Pools](#)
- Registerkarte Aktivität
 - Aktivitätsbenachrichtigungen — [Amazon SNS SNS-Benachrichtigungen](#)
- Registerkarte Überwachung
 - Auf dieser Registerkarte gibt es nur eine einzige Option, mit der Sie die [Erfassung von CloudWatch Gruppenmetriken](#) aktivieren oder deaktivieren können.

Aktualisieren Sie wie folgt eine Auto-Scaling-Gruppe über die Befehlszeile:

Verwenden Sie einen der folgenden Befehle:

- [update-auto-scaling-group](#) (AWS CLI)
- [Aktualisieren- ASAuto ScalingGroup](#) (AWS Tools for Windows PowerShell)

Aktualisieren von Auto-Scaling-Instances

Wenn Sie eine neue Startvorlage oder Startkonfiguration mit einer Auto-Scaling-Gruppe verknüpfen, erhalten alle neuen Instances die aktualisierte Konfiguration. Vorhandene Instances werden weiterhin mit der Konfiguration ausgeführt, mit der sie ursprünglich gestartet wurden. Um Ihre Änderungen auf vorhandene Instances anzuwenden, haben Sie die folgenden Möglichkeiten:

- Starten Sie eine Instance-Aktualisierung, um die älteren Instances zu ersetzen. Weitere Informationen finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).
- Warten Sie auf Skalierungsaktivitäten, um ältere Instances allmählich durch neuere Instances auf der Grundlage Ihrer [Beendigungsrichtlinien](#) zu ersetzen.
- Beenden Sie diese manuell, damit sie durch Ihre Auto-Scaling-Gruppe ersetzt werden.

Note

Sie können die folgenden Instance-Attribute ändern, indem Sie sie als Teil der Startvorlage oder Startkonfiguration angeben:

- Amazon Machine Image (AMI)
- Blockgeräte

- Schlüsselpaar
- Instance-Typ
- Sicherheitsgruppen
- Benutzerdaten
- Überwachung
- IAM-Instanzprofil
- Placement Tenancy
- kernel
- Ramdisk
- gibt an, ob Sie eine öffentliche IP-Adresse haben.
- die Verteilungsstrategie für die Availability Zone

Auto Scaling Scaling-Gruppenzuweisungsstrategie und Kapazitätsänderungen

Wenn Sie eine Auto Scaling Scaling-Gruppenzuweisungsstrategie ändern, werden vorhandene Instances nicht ersetzt. Alle neuen Instances, die aufgrund von Scale-Out-Ereignissen gestartet werden, folgen der neuen Zuweisungsstrategie. Jede future Skala von Ereignissen folgt den [Kündigungsrichtlinien](#) und verwendet die neue Zuweisungsstrategie, wenn die Kündigungsrichtlinie auf Default oder festgelegt ist `AllocationStrategy`. Wenn Sie beispielsweise die Zuweisungsstrategie von `lowest-price` auf `price-capacity-optimized` ändern, werden möglicherweise keine Instances beendet, aber alle neuen Instances werden mit der neuen Zuweisungsstrategie gestartet. Änderungen des Instance-Typs wirken sich nicht auf bestehende Instances aus.

Wenn Sie bestimmte Parameter wie den [OnDemandBaseCapacity](#) oder den ändern [OnDemandPercentageAboveBaseCapacity](#), führt Auto Scaling automatisch einen Neuausgleich durch, falls der Prozentsatz der On-Demand-Instances und Spot-Instances nicht den neuen Spezifikationen entspricht. Nehmen wir zum Beispiel an, eine Auto Scaling Scaling-Gruppe hat die `OnDemandPercentageAboveBaseCapacity` Einstellung auf 50 Prozent On-Demand-Instances und 50 Prozent Spot-Instances festgelegt. Dann `OnDemandPercentageAboveBaseCapacity` wird die Zahl auf 100 Prozent On-Demand-Instances erhöht. Die Auto Scaling Scaling-Gruppe wird proaktiv das Gleichgewicht wieder herstellen, indem sie neue On-Demand-Instances einführt und

Spot-Instances beendet. Die von Ihnen definierte [Wartungsrichtlinie für Instances](#) bestimmt die Reihenfolge der Start- und Kündigungsaktivitäten.

Tagging von Auto-Scaling-Gruppen und Instances

Ein Tag ist eine benutzerdefinierte Attributbezeichnung, die Sie zuweisen oder die einer AWS Ressource AWS zugewiesen wird. Jedes -Tag besteht aus zwei Teilen:

- einem Tag-Schlüssel (z. B. `costcenter`, `environment` oder `project`)
- einem optionalen Feld, dem sogenannten Tag-Wert (z. B. `111122223333` oder `production`)

Tags sind für folgende Aktivitäten nützlich:

- Verfolgen Sie Ihre AWS Kosten. Sie aktivieren diese Tags auf dem AWS Fakturierung und Kostenmanagement Dashboard. AWS verwendet die Tags, um Ihre Kosten zu kategorisieren und Ihnen einen monatlichen Kostenverteilungsbericht zu senden. Weitere Informationen finden Sie unter [Verwendung von Tags zur Kostenzuordnung](#) im Benutzerhandbuch zu AWS Billing .
- Steuern Sie den Zugriff auf Auto Scaling-Ressourcen basierend auf Tags. Sie können Bedingungen in Ihren IAM-Richtlinien zum Steuern des Zugriffs auf Auto-Scaling-Gruppen auf Basis der Tags für diese Gruppe verwenden. Weitere Informationen finden Sie unter [Tags für Sicherheit](#).
- Filtern und suchen Sie nach Auto-Scaling-Gruppen anhand der von Ihnen hinzugefügten Tags. Weitere Informationen finden Sie unter [Verwenden Sie Tags, um Auto-Scaling-Gruppen zu filtern](#).
- Identifizieren und organisieren Sie Ihre AWS Ressourcen. Viele AWS-Services unterstützen Tagging, sodass Sie Ressourcen aus verschiedenen Diensten dasselbe Tag zuweisen können, um anzuzeigen, dass die Ressourcen miteinander verknüpft sind.

Sie können neue oder vorhandene Auto-Scaling-Gruppen markieren. Sie können auch Tags aus einer Auto-Scaling-Gruppe an die von ihr gestarteten EC2-Instances weitergeben.

Tags werden nicht an Amazon EBS-Volumes verbreitet. Um Tags zu Amazon EBS-Volumes hinzuzufügen, geben Sie die Tags in einer Startvorlage an. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).

Sie können Tags mithilfe von AWS-Managementkonsole AWS CLI, oder SDKs erstellen und verwalten.

Inhalt

- [Einschränkungen für die Tag-Benennung und -Nutzung](#)
- [Tagging-Lebenszyklus von EC2-Instances](#)
- [Markieren Ihrer Auto-Scaling-Gruppen](#)
- [Löschen von Tags](#)
- [Tags für Sicherheit](#)
- [Steuern des Zugriffs auf Tags](#)
- [Verwenden Sie Tags, um Auto-Scaling-Gruppen zu filtern](#)

Einschränkungen für die Tag-Benennung und -Nutzung

Die folgenden grundlegenden Einschränkungen gelten für Tags (Markierungen):

- Die maximale Anzahl an Tags pro Ressource beträgt 50.
- Die maximale Anzahl an Tags, die Sie mit einem einzigen Aufruf hinzufügen oder entfernen können, beträgt 25.
- Die maximale Schlüssellänge beträgt 128 Unicode-Zeichen.
- Die maximale Wertlänge beträgt 256 Unicode-Zeichen.
- Bei Tag-Schlüsseln und -Werten wird zwischen Groß- und Kleinschreibung unterschieden. Eine bewährte Methode besteht darin, sich für eine einheitliche Schreibweise der Tag-Benennungen zu entscheiden und diese Strategie für alle Ressourcentypen umzusetzen.
- Verwenden Sie das `aws :` Präfix nicht in Ihren Tagnamen oder -Werten, da es für die AWS Verwendung reserviert ist. Sie können Tag-Namen oder -Werte mit diesem Präfix nicht bearbeiten oder löschen und sie werden nicht zu Ihren Tags pro Ressourcenkontingent gezählt.

Tagging-Lebenszyklus von EC2-Instances

Wenn Sie sich für die Weitergabe von Tags an Ihre EC2-Instances entschieden haben, werden die Tags wie folgt gehandhabt:

- Wenn eine Auto-Scaling-Gruppe Instances startet, fügt sie diesen während der Ressourcenerstellung Tags hinzu, nicht nach der Erstellung der Ressource.
- Die Auto-Scaling-Gruppe fügt den Instances automatisch einen Tag mit dem Schlüssel `aws : autoscaling : groupName` und einen Wert des Namens der Auto-Scaling-Gruppe hinzu.

- Wenn Sie Instance-Tags in Ihrer Startvorlage angeben und sich dafür entschieden haben, die Tags Ihrer Gruppe an ihre Instances zu übertragen, werden alle Tags zusammengeführt. Wenn derselbe Tag-Schlüssel für einen Tag in Ihrer Startvorlage und einen Tag in Ihrer Auto-Scaling-Gruppe angegeben wird, hat der Tag-Wert aus der Gruppe Vorrang.
- Beim Anfügen vorhandener Instances fügt die Auto-Scaling-Gruppe die Tags den Instances hinzu. Hierbei werden alle vorhandenen Tags mit demselben Tag-Schlüssel überschrieben. Zusätzlich fügt sie einen Tag mit `aws:autoscaling:groupName` als Schlüssel und mit dem Namen der Auto-Scaling-Gruppe als Wert hinzu.
- Beim Trennen einer Instance von einer Auto-Scaling-Gruppe entfernt sie nur das `aws:autoscaling:groupName`-Tag.

Markieren Ihrer Auto-Scaling-Gruppen

Beim Hinzufügen eines Tags zu einer Auto-Scaling-Gruppe können Sie angeben, ob es zu gestarteten Instances in der Auto-Scaling-Gruppe hinzugefügt werden soll. Nach der Änderung eines Tags wird neuen Instances in der Auto-Scaling-Gruppe die aktualisierte Version des Tags hinzugefügt. Bei der Erstellung oder Änderung eines Tags einer Auto-Scaling-Gruppe werden diese Änderungen an den bereits gestarteten Instances der Auto-Scaling-Gruppe nicht vorgenommen.

Inhalt

- [Hinzufügen oder Ändern von Tags \(Konsole\)](#)
- [Hinzufügen oder Ändern von Tags \(AWS CLI\)](#)

Hinzufügen oder Ändern von Tags (Konsole)

So markieren Sie eine Auto-Scaling-Gruppe bei der Erstellung

Wenn Sie eine Auto-Scaling-Gruppe mit der Amazon EC2-Konsole erstellen, können Sie auf der Seite Add Tags (Tags hinzufügen) im Assistenten zum Erstellen von Auto-Scaling-Gruppen Tag-Schlüssel und -Werte angeben. Zum Übertragen eines Tags an die in der Auto-Scaling-Gruppe gestarteten Instances achten Sie darauf, dass die Option Tag New Instances (Neue Instances markieren) für das Tag ausgewählt bleibt. Andernfalls können Sie sie deaktivieren.

Hinzufügen oder Ändern von Tags für eine vorhandene Auto-Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.

2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Tags, Bearbeiten.
4. Bearbeiten Sie zum Ändern bestehender Tags die Werte Key und Value.
5. Um ein neues Tag hinzuzufügen, wählen Sie Add tag aus und bearbeiten Sie Key und Value. Lassen Sie Tag new instances (Neue Instances markieren) aktiviert, damit das Tag zu in der Auto-Scaling-Gruppe gestarteten Instances automatisch hinzugefügt wird, oder deaktivieren Sie die Option, falls dies nicht erwünscht ist.
6. Wenn Sie mit dem Hinzufügen der Tags fertig sind, wählen Sie Update (Aktualisieren).

Hinzufügen oder Ändern von Tags (AWS CLI)

Die folgenden Beispiele zeigen, wie Sie Tags hinzufügen, wenn Sie Auto Scaling Scaling-Gruppen erstellen, und wie Sie Tags für bestehende Auto Scaling Scaling-Gruppen hinzufügen oder ändern können. AWS CLI

So markieren Sie eine Auto-Scaling-Gruppe bei der Erstellung

Verwenden Sie den [create-auto-scaling-group](#) Befehl, um eine neue Auto Scaling Scaling-Gruppe zu erstellen und der Auto Scaling Scaling-Gruppe beispielsweise ein Tag hinzuzufügen. **environment=production** Das Tag wird auch jeder gestarteten Instance in der Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
  --launch-configuration-name my-launch-config --min-size 1 --max-size 3 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
  --tags Key=environment,Value=production,PropagateAtLaunch=true
```

Erstellen oder Ändern von Tags für eine vorhandene Auto-Scaling-Gruppe

Verwenden Sie den [create-or-update-tags](#)-Befehl, um ein Tag zu erstellen oder zu ändern. Der folgende Befehl fügt z. B. die Tags **Name=my-asg** und **costcenter=cc123** hinzu. Die Tags werden nach dieser Änderung auch jeder gestarteten Instance in der Auto-Scaling-Gruppe hinzugefügt. Ist ein Tag mit einem dieser Schlüssel bereits vorhanden, wird das vorhandene Tag ersetzt. Die Amazon EC2-Konsole ordnet den Anzeigenamen für jede Instance dem Namen zu, der für den Name-Schlüssel angegeben ist (unter Beachtung der Groß-/Kleinschreibung).

```
aws autoscaling create-or-update-tags \
  --tags ResourceId=my-asg,ResourceType=auto-scaling-group,Key=Name,Value=my-
asg,PropagateAtLaunch=true \
  ResourceId=my-asg,ResourceType=auto-scaling-
  group,Key=costcenter,Value=cc123,PropagateAtLaunch=true
```

Beschreiben der Tags für eine Auto-Scaling-Gruppe (AWS CLI)

Wenn Sie die Tags anzeigen möchten, die auf eine bestimmte Auto-Scaling-Gruppe angewendet werden, können Sie entweder einen der folgenden Befehle verwenden:

- [describe-tags](#) — Sie geben Ihren Auto Scaling Scaling-Gruppennamen ein, um eine Liste der Tags für die angegebene Gruppe anzuzeigen.

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "Tags": [
    {
      "ResourceType": "auto-scaling-group",
      "ResourceId": "my-asg",
      "PropagateAtLaunch": true,
      "Value": "production",
      "Key": "environment"
    }
  ]
}
```

- [describe-auto-scaling-groups](#)— Sie geben Ihren Auto Scaling Scaling-Gruppennamen an, um die Attribute der angegebenen Gruppe, einschließlich aller Tags, anzuzeigen.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "AutoScalingGroups": [
```

```
{
  "AutoScalingGroupName": "my-asg",
  "AutoScalingGroupARN": "arn",
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
    "LaunchTemplateName": "my-launch-template",
    "Version": "$Latest"
  },
  "MinSize": 1,
  "MaxSize": 5,
  "DesiredCapacity": 1,
  ...
  "Tags": [
    {
      "ResourceType": "auto-scaling-group",
      "ResourceId": "my-asg",
      "PropagateAtLaunch": true,
      "Value": "production",
      "Key": "environment"
    }
  ],
  ...
}
]
```

Löschen von Tags

Sie können Tags, die einer Auto-Scaling-Gruppe zugeordnet sind, jederzeit löschen.

Inhalt

- [Löschen von Tags \(Konsole\)](#)
- [Löschen von Tags \(AWS CLI\)](#)

Löschen von Tags (Konsole)

So löschen Sie ein Tag

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.

2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Tags, Bearbeiten.
4. Wählen Sie Remove (Entfernen) neben dem Tag.
5. Wählen Sie Aktualisieren aus.

Löschen von Tags (AWS CLI)

Verwenden Sie den Befehl [delete-tags](#), um ein Tag zu löschen. Mit dem folgenden Befehl wird beispielsweise ein Tag mit dem Schlüssel **environment** gelöscht.

```
aws autoscaling delete-tags --tags "ResourceId=my-asg,ResourceType=auto-scaling-group,Key=environment"
```

Sie müssen den Tag-Schlüssel angeben, nicht aber den Wert. Wenn Sie einen Wert angeben und der Wert nicht korrekt ist, wird das Tag nicht gelöscht.

Tags für Sicherheit

Verwenden Sie Tags, um zu überprüfen, ob der Anforderer (z. B. ein IAM-Benutzer oder eine IAM-Rolle) über Berechtigungen zum Erstellen, Ändern oder Löschen bestimmter Auto-Scaling-Gruppen verfügt. Geben Sie Tag-Informationen im Bedingungelement einer IAM-Richtlinie mithilfe eines oder mehrerer der folgenden Bedingungsschlüssel an:

- Verwenden Sie `autoscaling:ResourceTag/tag-key: tag-value`, um Benutzeraktionen für Auto Scaling-Gruppen mit bestimmten Tags zuzulassen (oder zu verweigern).
- Schreiben Sie mit `aws:RequestTag/tag-key: tag-value` vor, dass in einer Anforderung ein bestimmtes Tag vorhanden (oder nicht vorhanden) sein muss.
- Schreiben Sie mit `aws:TagKeys [tag-key, ...]` vor, dass in einer Anforderung bestimmte Tag-Schlüssel vorhanden (oder nicht vorhanden) sein müssen.

Sie könnten beispielsweise den Zugriff auf alle Auto-Scaling-Gruppen, die ein Tag mit dem Schlüssel **environment** und dem Wert **production** enthalten, wie im folgenden Beispiel verweigern.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling>DeleteAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {"autoscaling:ResourceTag/environment":
          "production"}
      }
    }
  ]
}
```

Weitere Informationen über die Verwendung von Bedingungsschlüsseln zur Kontrolle des Zugriffs auf Auto-Scaling-Gruppen finden Sie unter [Funktionsweise von Amazon EC2 Auto Scaling mit IAM](#).

Steuern des Zugriffs auf Tags

Verwenden Sie Tags, um zu überprüfen, ob der Anforderer (z. B. ein IAM-Benutzer oder eine IAM-Rolle) über Berechtigungen zum Hinzufügen, Ändern oder Löschen von Tags für Auto-Scaling-Gruppen verfügt.

Die folgende Beispiel-IAM-Richtlinie gibt dem Prinzipal die Berechtigung, nur das Tag mit dem **temporary**-Schlüssel aus Auto-Scaling-Gruppen zu entfernen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
        "Effect": "Allow",
        "Action": "autoscaling:DeleteTags",
        "Resource": "*",
        "Condition": {
            "ForAllValues:StringEquals": { "aws:TagKeys": ["temporary"] }
        }
    }
]
```

Weitere Beispiele für IAM-Richtlinien, die Einschränkungen für die für Auto-Scaling-Gruppen angegebenen Tags erzwingen, finden Sie unter [Steuern, welche Tag-Schlüssel und Tag-Werte verwendet werden können](#).

Note

Selbst wenn Sie eine Richtlinie haben, die Ihre Benutzer daran hindert, einen Tagging-Vorgang für eine Auto-Scaling-Gruppe durchzuführen (oder rückgängig zu machen), bedeutet dies nicht, dass sie die Tags der Instances nach dem Start manuell ändern können. Beispiele zur Steuerung des Zugriffs auf Tags auf EC2-Instances finden Sie unter [Beispiel: Tagging resources](#) im Amazon EC2 EC2-Benutzerhandbuch.

Verwenden Sie Tags, um Auto-Scaling-Gruppen zu filtern

Die folgenden Beispiele zeigen Ihnen, wie Sie Filter mit dem [describe-auto-scaling-groups](#) Befehl verwenden, um Auto Scaling Scaling-Gruppen mit bestimmten Tags zu beschreiben. Das Filtern nach Tags ist auf das AWS CLI oder ein SDK beschränkt und in der Konsole nicht verfügbar.

Überlegungen zum Filtern

- Sie können mehrere Filter und mehrere Filterwerte in einer einzelnen Anforderung angeben.
- Sie können Platzhalter in den Filterwerten nicht verwenden.
- Bei Filterwerten muss die Groß- und Kleinschreibung beachtet werden.

Beispiel: Beschreiben Sie Auto-Scaling-Gruppen mit einem bestimmten Tag-Schlüssel und Wertepaar

Der folgende Befehl zeigt, wie Sie die Ergebnisse so filtern, dass nur Auto-Scaling-Gruppen mit dem Tag-Schlüssel und dem Wertepaar **environment=production** angezeigt werden.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-value,Values=production
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn",
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
      "MinSize": 1,
      "MaxSize": 5,
      "DesiredCapacity": 1,
      "...",
      "Tags": [
        {
          "ResourceType": "auto-scaling-group",
          "ResourceId": "my-asg",
          "PropagateAtLaunch": true,
          "Value": "production",
          "Key": "environment"
        }
      ],
      "...",
    },
    ... additional groups ...
  ]
}
```

Alternativ können Sie auch Tags mit einem `tag:<key>`-Filter angeben. Der folgende Befehl zeigt zum Beispiel, wie Sie die Ergebnisse filtern können, um nur Auto-Scaling-Gruppen mit dem Tag-Schlüssel und dem Wertepaar **environment=production** anzuzeigen. Dieser Filter ist wie folgt

formatiert: Name=tag:<key>, Values=<value>, wobei <key> und <value> ein Tag-Schlüssel- und Wertepaar darstellen.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag:environment,Values=production
```

Sie können die AWS CLI Ausgabe auch mithilfe der `--query` Option filtern. Das folgende Beispiel zeigt, wie die AWS CLI Ausgabe für den vorherigen Befehl nur auf den Gruppennamen, die Mindestgröße, die Maximalgröße und die gewünschten Kapazitätsattribute beschränkt werden kann.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag:environment,Values=production \  
  --query "AutoScalingGroups[].{AutoScalingGroupName: AutoScalingGroupName, MinSize: MinSize, MaxSize: MaxSize, DesiredCapacity: DesiredCapacity}"
```

Nachfolgend finden Sie eine Beispielantwort.

```
[  
  {  
    "AutoScalingGroupName": "my-asg",  
    "MinSize": 0,  
    "MaxSize": 10,  
    "DesiredCapacity": 1  
  },  
  ... additional groups ...  
]
```

Weitere Informationen zum Filtern finden Sie im AWS Command Line Interface Benutzerhandbuch unter [Filtern der AWS CLI Ausgabe](#).

Beispiel: Beschreiben Sie Auto-Scaling-Gruppen mit Tags, die mit dem angegebenen Tag-Schlüssel übereinstimmen

Der folgende Befehl zeigt, wie Sie die Ergebnisse so filtern, dass nur Auto-Scaling-Gruppen mit dem Tag angezeigt werden, unabhängig vom Wert des **environment**-Tags.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag-key,Values=environment
```

Beispiel: Beschreiben Sie Auto-Scaling-Gruppen mit Tags, die dem angegebenen Satz von Tag-Schlüsseln entsprechen

Der folgende Befehl zeigt, wie Sie die Ergebnisse so filtern, dass nur Auto-Skalierungsgruppen mit Tags für **environment** und **project** angezeigt werden, unabhängig von den Tag-Werten.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag-key,Values=environment Name=tag-key,Values=project
```

Beispiel: Beschreiben Sie Auto-Scaling-Gruppen mit Tags, die mindestens einem der angegebenen Tag-Schlüssel entsprechen

Der folgende Befehl zeigt, wie Sie die Ergebnisse so filtern, dass nur Auto-Scaling-Gruppen mit Tags für **environment** oder **project** angezeigt werden, unabhängig von den Tag-Werten.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag-key,Values=environment,project
```

Beispiel: Beschreiben Sie Auto-Scaling-Gruppen mit dem angegebenen Tag-Wert

Der folgende Befehl zeigt, wie Sie die Ergebnisse so filtern, dass nur Auto-Scaling-Gruppen mit einem Tag-Wert von **production** angezeigt werden, unabhängig vom Tag-Schlüssel.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag-value,Values=production
```

Beispiel: Beschreiben Sie Auto-Scaling-Gruppen mit den angegebenen Tag-Werten

Der folgende Befehl zeigt, wie Sie die Ergebnisse so filtern, dass nur Auto-Scaling-Gruppen mit den Tag-Werten **production** und **development** angezeigt werden, unabhängig vom Tagschlüssel.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag-value,Values=production Name=tag-value,Values=development
```

Beispiel: Beschreiben Sie Auto-Scaling-Gruppen mit Tags, die mindestens einem der angegebenen Tag-Werte entsprechen

Der folgende Befehl zeigt, wie Sie die Ergebnisse filtern können, um nur Auto-Scaling-Gruppen mit einem Tag-Wert von **production** oder **development** anzuzeigen, unabhängig vom Tag-Schlüssel.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag-value,Values=production,development
```

```
--filters Name=tag-value,Values=production,development
```

Beispiel: Beschreiben Sie Auto-Scaling-Gruppen mit Tags, die mehreren Tag-Schlüsseln und Werten entsprechen

Sie können auch Filter kombinieren, um benutzerdefinierte AND- und OR-Logik zu erstellen und so eine komplexere Filterung durchzuführen.

Der folgende Befehl zeigt, wie Sie die Ergebnisse filtern können, um nur Auto-Scaling-Gruppen mit einer bestimmten Gruppe von Tags anzuzeigen. Ein Tag-Schlüssel ist **environment** AND der Tag-Wert ist (**production** OR **development**) AND der andere Tag-Schlüssel ist **costcenter** AND der Tag-Wert ist **cc123**.

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag:environment,Values=production,development \  
  Name=tag:costcenter,Values=cc123
```

Wartungsrichtlinien für Instances

Sie können eine Instance-Wartungsrichtlinie für Ihre Auto-Scaling-Gruppe konfigurieren, um bestimmte Kapazitätsanforderungen bei Ereignissen zu erfüllen, die dazu führen, dass Instances ersetzt werden, z. B. bei einer Instance-Aktualisierung oder bei der Zustandsprüfung.

Nehmen wir an, Sie verfügen über eine Auto-Scaling-Gruppe, die eine geringe Anzahl von Instances aufweist. Sie möchten vermeiden, dass es zu möglichen Störungen kommt, wenn eine Instance beendet und anschließend ersetzt wird, sobald Zustandsprüfungen auf eine beeinträchtigte Instance hinweisen. Mit einer Instance-Wartungsrichtlinie können Sie sicherstellen, dass Amazon EC2 Auto Scaling zuerst eine neue Instance startet und dann wartet, bis sie vollständig bereit ist, bevor es die fehlerhafte Instance beendet.

Eine Instance-Wartungsrichtlinie hilft Ihnen auch dabei, mögliche Störungen zu minimieren, wenn mehrere Instances gleichzeitig ersetzt werden. Sie legen die minimalen und maximalen gesunden Prozentwerte für die Richtlinie fest, und Ihre Auto-Scaling-Gruppe kann die Kapazität nur innerhalb dieses minimalen und maximalen Bereichs erhöhen und verringern, wenn Instances ersetzt werden. Ein größerer Bereich erhöht die Anzahl der Instances, die gleichzeitig ausgetauscht werden können.

Inhalt

- [Instanzwartungsrichtlinie für die Auto Scaling Scaling-Gruppe](#)
- [Festlegen einer Instance-Wartungsrichtlinie für Ihre Auto-Scaling-Gruppe](#)

Instandwartungsrichtlinie für die Auto Scaling Scaling-Gruppe

Dieses Thema bietet einen Überblick über die verfügbaren Optionen und beschreibt, was zu beachten ist, wenn Sie eine Instance-Wartungsrichtlinie erstellen.

Inhalt

- [-Übersicht](#)
- [Schlüsselkonzepte](#)
- [Instance-Aufwärmphase](#)
- [Frist der Zustandsprüfung](#)
- [Skalieren Ihrer Auto-Scaling-Gruppe](#)
- [Beispielszenarien](#)

-Übersicht

Wenn Sie eine Instance-Wartungsrichtlinie für Ihre Auto-Scaling-Gruppe erstellen, wirkt sich die Richtlinie auf die Ergebnisse von Amazon EC2 Auto Scaling aus, die dazu führen, dass Instances ersetzt werden. Dies führt zu einem konsistenteren Austauschverhalten innerhalb derselben Auto-Scaling-Gruppe. Außerdem können Sie Verfügbarkeit oder Kosten für Ihre Gruppe je nach Bedarf optimieren.

In der Konsole stehen die folgenden Konfigurationsoptionen zur Verfügung:

- Vor dem Beenden starten – Eine neue Instance muss zuerst bereitgestellt werden, bevor eine bestehende Instance beendet werden kann. Dieser Ansatz ist eine gute Wahl für Anwendungen, bei denen Verfügbarkeit wichtiger ist als Kosteneinsparungen.
- Beenden und starten – Neue Instances werden zur gleichen Zeit bereitgestellt, wie Ihre bestehenden Instances beendet werden. Dieser Ansatz ist eine gute Wahl für Anwendungen, bei denen Kosteneinsparungen wichtiger sind als die Verfügbarkeit. Es ist auch eine gute Wahl für Anwendungen, die nicht mehr Kapazität benötigen, als derzeit verfügbar ist, selbst wenn Instances ersetzt werden.
- Benutzerdefinierte Richtlinie – Mit dieser Option können Sie für Ihre Richtlinie einen benutzerdefinierten Mindest- und Höchstbereich für die Kapazität einrichten, die beim Austausch von Instances verfügbar sein soll. Dieser Ansatz kann Ihnen helfen, das richtige Gleichgewicht zwischen Kosten und Verfügbarkeit zu finden.

Die Standardeinstellung für eine Auto-Scaling-Gruppe ist, dass sie keine Instance-Wartungsrichtlinie hat, was dazu führt, dass sie auf Instance-Wartungsereignisse mit dem Standardverhalten reagiert. Das Standardverhalten wird in der folgenden Tabelle beschrieben.

Standardverhalten bei Instance-Wartungsereignissen

Veranstaltung	Description	Standardverhalten
Fehlgeschlagene Zustandsprüfung	Passiert automatisch, wenn Instances ihre Zustandsprüfungen nicht bestehen. Amazon EC2 Auto Scaling ersetzt die Instances, die ihre Zustandsprüfungen nicht bestehen. Informationen zu den Ursachen für fehlgeschlagene Zustandsprüfungen finden Sie unter Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe .	Beenden und starten.
Instance-Aktualisierung	Das geschieht, wenn Sie eine Instance-Aktualisierung starten. Abhängig von Ihrer Konfiguration kann eine Instance-Aktualisierung eine einzelne Instance, mehrere Instances auf einmal oder alle auf einmal ersetzen. Weitere Informationen finden Sie unter Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren .	Beenden und starten.
Maximale Lebensdauer von Instances	Das passiert automatisch, wenn Instances die maximale Instance-Lebensdauer	Beenden und starten.

Veranstaltung	Description	Standardverhalten
	<p>er erreichen, die Sie für Ihre Auto-Scaling-Gruppe angegeben haben. Amazon EC2 Auto Scaling ersetzt solche Instances, die ihre maximale Instance-Lebensdauer erreichen. Weitere Informationen finden Sie unter Auto-Scaling-Instances basierend auf der maximalen Instance-Lebensdauer ersetzen.</p>	

Veranstaltung	Description	Standardverhalten
Neuausgleich	<p>Dies erfolgt automatisch, wenn grundlegende Änderungen vorliegen, die dazu führen, dass die Gruppe aus dem Gleichgewicht gerät. Amazon EC2 Auto Scaling führt in den folgenden Situationen einen Neuausgleich für die Gruppe aus:</p> <ul style="list-style-type: none">• Eine Availability Zone, die zuvor zu wenig Kapazität hatte, wurde wiederhergestellt, oder Sie fügen der Gruppe eine Availability Zone hinzu oder entfernen sie aus ihr. In diesem Fall versucht Ihre Auto-Scaling-Gruppe, sich gleichmäßig über die Availability Zones zu verteilen. Weitere Informationen finden Sie unter Wiederherstellen des Gleichgewichts von Aktivitäten.• Sie aktivieren den Kapazitätsausgleich in Ihrer Auto-Scaling-Gruppe, und sie versucht, neue Spot Instances zu starten, bevor vorhandene unterbrochen werden, wenn sich die Verfügbarkeit von Spot Instances ändert. Weitere Informationen finden Sie	<p>Vor dem Beenden starten.</p> <p>Amazon EC2 Auto Scaling kann die Größenlimits Ihrer Gruppe um bis zu 10 Prozent der maximalen Kapazität überschreiten. Wenn Sie den Kapazitätsausgleich verwenden, können diese Grenzwerte jedoch nur um bis zu 10 Prozent der gewünschten Kapazität überschritten werden.</p>

Veranstaltung	Description	Standardverhalten
	<p>unter Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances.</p> <ul style="list-style-type: none"> Sie aktualisieren Ihre Auto-Scaling-Gruppe und sie ersetzt nach und nach Instances, um sie an die neuen Kaufoptionen anzupassen, die Sie bei der Aktualisierung einer Richtlinie für gemischte Instances ausgewählt haben. Weitere Informationen finden Sie unter Aktualisieren einer Auto-Scaling-Gruppe. 	

Amazon EC2 Auto Scaling wird in den folgenden Situationen weiterhin standardmäßig beendet und gestartet. Wenn eine dieser Situationen eintritt, liegt die Kapazität Ihrer Gruppe daher u. U. unter dem unteren Schwellenwert Ihrer Instance-Wartungsrichtlinie.

- Wenn eine Instance unerwartet beendet wird, z. B. aufgrund menschlichen Eingreifens. Amazon EC2 Auto Scaling ersetzt sofort Instances, die nicht mehr ausgeführt werden. Weitere Informationen finden Sie unter [Zustandsprüfungen von Amazon EC2](#).
- Wenn Amazon EC2 eine Instance im Rahmen eines geplanten Ereignisses neu startet, stoppt oder außer Betrieb setzt, bevor Amazon EC2 Auto Scaling die Ersatz-Instance starten kann. Weitere Informationen zu diesen Ereignissen finden Sie unter [Geplante Ereignisse für Ihre Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Wenn der Amazon EC2 Spot Service eine Spot-Instance-Unterbrechung einleitet und anschließend das Beenden einer Spot Instance erzwungen wird.

Wenn Sie bei Spot Instances den Kapazitätsausgleich in Ihrer Auto-Scaling-Gruppe aktiviert haben, kann es sein, dass die Instance bereits eine anhängige Instance aus einem anderen Spot-Pool hat,

die gestartet wurde, bevor die Spot-Unterbrechung eingeleitet wurde. Weitere Informationen darüber, wie der Kapazitätsausgleich funktioniert, finden Sie unter [Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances](#).

Da jedoch nicht garantiert werden kann, dass Spot Instances verfügbar bleiben, und sie mit einer zweiminütigen Benachrichtigung über die Unterbrechung der Spot Instance beendet werden können, kann der untere Schwellenwert Ihrer Instance-Wartungsrichtlinie überschritten werden, wenn Instances unterbrochen werden, bevor Ihre neuen Instances gestartet wurden.

Schlüsselkonzepte

Bevor Sie beginnen, sollten Sie sich mit folgenden Kernkonzepten und der Terminologie vertraut machen:

Gewünschte Kapazität

Die gewünschte Kapazität stellt die Kapazität der Auto-Scaling-Gruppe zum Zeitpunkt der Erstellung dar. Dies ist auch die Kapazität, die die Gruppe aufrechtzuerhalten versucht, wenn keine Skalierungsbedingungen an die Gruppe angehängt wurden.

Instance-Wartungsrichtlinie

Eine Instance-Wartungsrichtlinie steuert, ob eine Instance zuerst bereitgestellt wird, bevor eine vorhandene Instance wegen eines Instance-Wartungsereignisses beendet wird. Sie bestimmt auch, wie weit Ihre Auto-Scaling-Gruppe Ihre gewünschte Kapazität unterschreiten und überschreiten kann, um mehrere Instances gleichzeitig zu ersetzen.

Maximaler fehlerfreier Prozentsatz

Der maximale fehlerfreie Prozentsatz ist der Prozentsatz der gewünschten Kapazität, auf den Ihre Auto-Scaling-Gruppe beim Austausch von Instances erhöhen kann. Dies stellt den maximalen Prozentsatz der Gruppe dar, der zur Unterstützung Ihrer Workload in Betrieb und fehlerfrei oder ausstehend sein kann. In der Konsole können Sie den maximalen fehlerfreien Prozentsatz festlegen, wenn Sie entweder die Option Vor dem Beenden starten oder Benutzerdefinierte Richtlinie verwenden. Die gültigen Werte lauten 100–200 Prozent.

Minimaler fehlerfreier Prozentsatz

Der minimale fehlerfreie Prozentsatz ist der Prozentsatz der gewünschten Kapazität, die beim Austausch von Instances betriebsbereit, fehlerfrei und einsatzbereit zur Unterstützung Ihrer Arbeitslast bleiben soll. Eine Instance gilt als fehlerfrei und einsatzbereit, wenn sie ihren ersten Integritätstest erfolgreich abgeschlossen hat und die angegebene Aufwärmzeit verstrichen ist. In

der Konsole können Sie den minimalen fehlerfreien Prozentsatz festlegen, wenn Sie entweder die Option Beenden und starten oder Benutzerdefinierte Richtlinie verwenden. Die gültigen Werte lauten 0–100 Prozent.

Note

Um Instances schneller zu ersetzen, können Sie einen niedrigen Wert für den minimalen fehlerfreien Prozentsatz angeben. Wenn jedoch nicht genügend fehlerfreie Instanzen laufen, kann die Verfügbarkeit verringert werden. Wir empfehlen, einen angemessenen Wert auszuwählen, um die Verfügbarkeit in Situationen aufrechtzuerhalten, in denen mehrere Instances ersetzt werden.

Instance-Aufwärmphase

Wenn Ihre Instances nach dem Eintritt in den Status `InService` Zeit für die Initialisierung benötigen, aktivieren Sie die standardmäßige Instance-Aufwärmphase für Ihre Auto-Scaling-Gruppe. Mit der standardmäßigen Instance-Aufwärmphase können Sie verhindern, dass Instances auf den minimalen fehlerfreien Prozentsatz angerechnet werden, bevor sie bereit sind. Dadurch wird sichergestellt, dass Amazon EC2 Auto Scaling berücksichtigt, wie lange es dauert, bis genügend Kapazität zur Unterstützung der Workload vorhanden ist, bevor vorhandene Instances beendet werden.

Als zusätzlichen Vorteil können Sie die CloudWatch Amazon-Metriken, die für die dynamische Skalierung verwendet werden, verbessern, wenn Sie das Standard-Instance-Warmup aktivieren. Wenn Ihre Auto Scaling Scaling-Gruppe über Skalierungsrichtlinien verfügt, verwendet sie beim Skalieren der Gruppe dieselbe Standard-Aufwärmphase, um zu verhindern, dass Instances auf die CloudWatch Metriken angerechnet werden, bevor die Initialisierung abgeschlossen ist.

Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

Frist der Zustandsprüfung

Amazon EC2 Auto Scaling bestimmt anhand des Status der von Ihrer Auto-Scaling-Gruppe verwendeten Zustandsprüfungen, ob eine Instance fehlerfrei ist. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Um sicherzustellen, dass diese Zustandsprüfungen so schnell wie möglich beginnen, sollten Sie die Karenzzeit für die Zustandsprüfung der Gruppe nicht zu hoch ansetzen, nur hoch genug, damit

Ihre Elastic Load Balancing-Zustandsprüfungen feststellen können, ob ein Ziel zur Bearbeitung von Anfragen verfügbar ist. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).

Skalieren Ihrer Auto-Scaling-Gruppe

Eine Instance-Wartungsrichtlinie gilt nur für Instance-Wartungsereignisse und verhindert nicht die manuelle oder automatische Skalierung der Gruppe.

Wenn Ihrer Auto-Scaling-Gruppe Skalierungsrichtlinien oder geplante Aktionen zugeordnet sind, können diese parallel ausgeführt werden, während Wartungsereignisse für Instances stattfinden. In einem solchen Fall könnten sie die gewünschte Kapazität der Gruppe erhöhen oder verringern, jedoch nur innerhalb der von Ihnen definierten Skalierungslimits. Weitere Informationen zu den Limits finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).

Beispielszenarien

In einem typischen Szenario könnten Ihre Instance-Wartungsrichtlinie und die gewünschte Kapazität ungefähr so aussehen:

- Minimaler fehlerfreier Prozentsatz = 90 Prozent
- Maximaler fehlerfreier Prozentsatz = 120 Prozent
- Gewünschte Kapazität = 100

Während eines Instance-Wartungsereignisses kann Ihre Auto-Scaling-Gruppe über 90 bis 120 Instances verfügen. Nach dem Ereignis verfügt die Gruppe wieder über 100 Instances.

Wenn Sie eine Instance-Wartungsrichtlinie mit einer Auto-Scaling-Gruppe verwenden, die über einen warmen Pool verfügt, werden die minimalen und maximalen fehlerfreien Prozentsätze getrennt auf die Auto-Scaling-Gruppe und den warmen Pool angewendet.

Nehmen wir die folgende Konfiguration als Beispiel:

- Minimaler fehlerfreier Prozentsatz = 90 Prozent
- Maximaler fehlerfreier Prozentsatz = 120 Prozent
- Gewünschte Kapazität = 100
- Größe des warmen Pools = 10

Wenn Sie eine Instance-Aktualisierung starten, um die Instances der Gruppe zu recyceln, ersetzt Amazon EC2 Auto Scaling zuerst die Instances in der Auto-Scaling-Gruppe und dann die Instances im warmen Pool. Amazon EC2 Auto Scaling arbeitet zwar immer noch daran, Instances in der Auto-Scaling-Gruppe zu ersetzen, aber die Gruppe könnte zwischen 90 und 120 Instances haben. Nach Fertigstellung der Gruppe kann Amazon EC2 Auto Scaling daran arbeiten, Instances im warmen Pool zu ersetzen. Währenddessen kann der warme Pool zwischen 9 und 12 Instances haben.

Festlegen einer Instance-Wartungsrichtlinie für Ihre Auto-Scaling-Gruppe

Sie können eine Instance-Wartungsrichtlinie erstellen, wenn Sie eine Auto-Scaling-Gruppe erstellen. Sie können sie auch für vorhandene Gruppen erstellen.

Durch Festlegen einer Instance-Wartungsrichtlinie für Ihre Auto-Scaling-Gruppe müssen Sie für die Instance-Aktualisierung keine Werte mehr angeben, es sei denn, Sie möchten die Instance-Wartungsrichtlinie überschreiben.

In der Konsole bietet Amazon EC2 Auto Scaling Optionen, die Ihnen die ersten Schritte erleichtern.

Inhalt

- [Festlegen einer Instance-Wartungsrichtlinie](#)
- [Entfernen einer Instance-Wartungsrichtlinie](#)

Festlegen einer Instance-Wartungsrichtlinie

Verwenden Sie eine der folgenden Methoden, um eine Instance-Wartungsrichtlinie für eine Auto-Scaling-Gruppe festzulegen:

Console

Festlegen einer Instance-Wartungsrichtlinie für eine Gruppe (Konsole)

1. Befolgen Sie die Anweisungen in [Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage](#) und führen Sie jeden Schritt des Verfahrens bis zu Schritt 11 durch.
2. Geben Sie unter Konfigurieren von Gruppengröße und Skalierungsrichtlinien für Gewünschte Kapazität die anfängliche Anzahl von Instances ein, die gestartet werden sollen.
3. Wenn im Abschnitt Skalierung unter Skalierungslimits Ihr neuer Wert für die gewünschte Kapazität größer als die gewünschte Mindestkapazität und die gewünschte Höchstkapazität ist, wird die gewünschte Höchstkapazität automatisch auf den neuen Wert für die gewünschte Kapazität erhöht. Sie können die Limits bei Bedarf ändern.

4. Wählen Sie für Automatische Skalierung aus, ob Sie eine Skalierungsrichtlinie für die Zielverfolgung erstellen möchten. Sie können diese Richtlinie auch erstellen, nachdem Sie Ihre Auto-Scaling-Gruppe erstellt haben.

Wenn Sie sich für die Skalierungsrichtlinie für die Zielverfolgung entscheiden, befolgen Sie die Anweisungen unter [Erstellen einer Zielverfolgungs-Skalierungsrichtlinie](#), um die Richtlinie zu erstellen.

5. Wählen Sie im Abschnitt Instance-Wartungsrichtlinie eine der verfügbaren Optionen aus:
 - Vor dem Beenden starten: Eine neue Instance muss zuerst bereitgestellt werden, bevor eine bestehende Instance beendet werden kann. Dies ist eine gute Wahl für Anwendungen, bei denen Verfügbarkeit wichtiger ist als Kosteneinsparungen.
 - Beenden und starten: Neue Instances werden zur gleichen Zeit bereitgestellt, wie Ihre bestehenden Instances beendet werden. Dies ist eine gute Wahl für Anwendungen, bei denen Kosteneinsparungen Vorrang vor der Verfügbarkeit haben. Es ist auch eine gute Wahl für Anwendungen, die nicht mehr Kapazität benötigen, als derzeit verfügbar ist.
 - Benutzerdefinierte Richtlinie: Mit dieser Option können Sie für Ihre Richtlinie einen benutzerdefinierten Mindest- und Höchstbereich für die Kapazität einrichten, die beim Austausch von Instances verfügbar sein soll. Dies kann Ihnen helfen, das richtige Gleichgewicht zwischen Kosten und Verfügbarkeit zu finden.
6. Geben Sie unter Fehlerfreien Prozentsatz festlegen Werte für eines oder beide der folgenden Felder ein. Die aktivierten Felder variieren je nach der Option, die Sie im vorherigen Schritt ausgewählt haben.
 - Min.: Legt den fehlerfreien Mindestprozentsatz fest, der erforderlich ist, um mit dem Ersetzen von Instances fortzufahren.
 - Max.: Legt den maximalen fehlerfreien Prozentsatz fest, der während des Ersetzens von Instances möglich ist.
7. Erweitern Sie den Abschnitt Kapazität bei Ersatz auf Grundlage Ihrer gewünschten Kapazität anzeigen, um zu überprüfen, ob die Werte für Min. und Max für Ihre Gruppe gelten. Welche genauen Werte verwendet werden, hängt vom gewünschten Kapazitätswert ab, der sich ändert, wenn die Gruppe skaliert wird.
8. Fahren Sie mit den Schritten unter [Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage](#) fort.

AWS CLI

Festlegen einer Instance-Wartungsrichtlinie für eine Gruppe (AWS CLI)

Fügen Sie dem Befehl die `--instance-maintenance-policy` Option hinzu. [create-auto-scaling-group](#) Im folgenden Beispiel wird eine Instance-Wartungsrichtlinie für eine neue Auto-Scaling-Gruppe mit dem Namen `my-asg` festgelegt.

```
aws autoscaling create-auto-scaling-group \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --auto-scaling-group-name my-asg \
  --min-size 1 \
  --max-size 10 \
  --desired-capacity 5 \
  --default-instance-warmup 20 \
  --instance-maintenance-policy '{
    "MinHealthyPercentage": 90,
    "MaxHealthyPercentage": 120
  }' \
  --vpc-zone-identifier "subnet-5e6example,subnet-613example,subnet-c93example"
```

Console

Festlegen einer Instance-Wartungsrichtlinie für eine vorhandene Gruppe (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die AWS-Region aus, in der Sie Ihre Auto-Scaling-Gruppe erstellt haben.
3. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Wählen Sie auf der Registerkarte Details die Option Instance-Wartungsrichtlinie, Bearbeiten aus.
5. Wählen Sie zum Festlegen einer Instance-Wartungsrichtlinie für die Gruppe eine der verfügbaren Optionen aus:

- Vor dem Beenden starten: Eine neue Instance muss zuerst bereitgestellt werden, bevor eine bestehende Instance beendet werden kann. Dies ist eine gute Wahl für Anwendungen, bei denen Verfügbarkeit wichtiger ist als Kosteneinsparungen.
 - Beenden und starten: Neue Instances werden zur gleichen Zeit bereitgestellt, wie Ihre bestehenden Instances beendet werden. Dies ist eine gute Wahl für Anwendungen, bei denen Kosteneinsparungen Vorrang vor der Verfügbarkeit haben. Es ist auch eine gute Wahl für Anwendungen, die nicht mehr Kapazität benötigen, als derzeit verfügbar ist.
 - Benutzerdefinierte Richtlinie: Mit dieser Option können Sie für Ihre Richtlinie einen benutzerdefinierten Mindest- und Höchstbereich für die Kapazität einrichten, die beim Austausch von Instances verfügbar sein soll. Dies kann Ihnen helfen, das richtige Gleichgewicht zwischen Kosten und Verfügbarkeit zu finden.
6. Geben Sie unter Fehlerfreien Prozentsatz festlegen Werte für eines oder beide der folgenden Felder ein. Die aktivierten Felder variieren je nach der Option, die Sie im vorherigen Schritt ausgewählt haben.
 - Min.: Legt den fehlerfreien Mindestprozentsatz fest, der erforderlich ist, um mit dem Ersetzen von Instances fortzufahren.
 - Max.: Legt den maximalen fehlerfreien Prozentsatz fest, der während des Ersetzens von Instances möglich ist.
 7. Erweitern Sie den Abschnitt Kapazität bei Ersatz auf Grundlage Ihrer gewünschten Kapazität anzeigen, um zu überprüfen, ob die Werte für Min. und Max für Ihre Gruppe gelten. Welche genauen Werte verwendet werden, hängt vom gewünschten Kapazitätswert ab, der sich ändert, wenn die Gruppe skaliert wird.
 8. Wählen Sie Aktualisieren aus.

AWS CLI

Festlegen einer Instance-Wartungsrichtlinie für eine vorhandene Gruppe (AWS CLI)

Fügen Sie die `--instance-maintenance-policy` Option zum [update-auto-scaling-group](#) Befehl hinzu. Im folgenden Beispiel wird eine Instance-Wartungsrichtlinie für eine spezifizierte Auto-Scaling-Gruppe festgelegt.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--instance-maintenance-policy '{  
  "MinHealthyPercentage": 90,}
```

```
"MaxHealthyPercentage": 120  
'
```

Entfernen einer Instance-Wartungsrichtlinie

Wenn Sie die Verwendung einer Instance-Wartungsrichtlinie mit Ihrer Auto-Scaling-Gruppe beenden möchten, können Sie sie entfernen.

Console

Entfernen einer Instance-Wartungsrichtlinie (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die AWS-Region aus, in der Sie Ihre Auto-Scaling-Gruppe erstellt haben.
3. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Wählen Sie auf der Registerkarte Details die Option Instance-Wartungsrichtlinie, Bearbeiten aus.
5. Wählen Sie Keine Instance-Wartungsrichtlinie aus.
6. Wählen Sie Aktualisieren aus.

AWS CLI

Entfernen einer Instance-Wartungsrichtlinie (AWS CLI)

Fügen Sie die `--instance-maintenance-policy` Option zum [update-auto-scaling-group](#) Befehl hinzu. Im folgenden Beispiel wird eine Instance-Wartungsrichtlinie einer spezifizierten Auto-Scaling-Gruppe entfernt.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--instance-maintenance-policy '{  
    "MinHealthyPercentage": -1,  
    "MaxHealthyPercentage": -1  
'
```

Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling bietet die Möglichkeit, Lebenszyklus-Hooks zu Ihren Auto-Scaling-Gruppen hinzuzufügen. Mit diesen Hooks können Sie Lösungen erstellen, die Ereignisse im Lebenszyklus von Auto-Scaling-Instances erkennen und dann eine benutzerdefinierte Aktion auf Instances ausführen, wenn das entsprechende Lebenszyklusereignis eintritt. Ein Lebenszyklus-Hook gibt eine bestimmte Zeitspanne vor (standardmäßig eine Stunde), in der auf den Abschluss der Aktion gewartet wird, bevor die Instance in den nächsten Zustand übergeht.

Als Beispiel für die Verwendung von Lebenszyklus-Hooks mit Auto-Scaling-Instances:

- Wenn ein horizontales Skalierungsereignis auftritt, schließt die neu gestartete Instance ihre Startsequenz ab und wechselt in einen Wartezustand. Während sich die Instance in einem Wartestatus befindet, können Sie auf ein Skript ausführen, um die erforderlichen Softwarepakete für Ihre Anwendung herunterzuladen und zu installieren. Stellen Sie dabei sicher, dass Ihre Instance vollständig bereit ist, bevor sie mit dem Empfang von Datenverkehr beginnt. Wenn das Skript die Installation der Software abgeschlossen hat, sendet es den `complete-lifecycle-action`-Befehl, um fortzufahren.
- Wenn ein Scale-In-Ereignis eintritt, pausiert ein Lifecycle-Hook die Instance, bevor sie beendet wird, und sendet Ihnen eine Benachrichtigung über Amazon. EventBridge Während sich die Instance im Wartestatus befindet, können Sie eine AWS Lambda Funktion aufrufen oder eine Verbindung zur Instance herstellen, um Protokolle oder andere Daten herunterzuladen, bevor die Instance vollständig beendet wird.

Eine beliebte Verwendung von Lebenszyklus-Hooks besteht darin, zu steuern, wann Instances bei Elastic Load Balancing registriert werden. Wenn Sie Ihrer Auto-Scaling-Gruppe einen Start-Lebenszyklus-Hook hinzufügen, können Sie sicherstellen, dass Ihre Bootstrap-Skripte erfolgreich abgeschlossen wurden und die Anwendungen auf den Instances bereit sind, Datenverkehr anzunehmen, bevor sie am Ende des Lebenszyklus-Hooks beim Load Balancer registriert werden.

Inhalt

- [Verfügbarkeit von Lebenszyklus-Hooks](#)
- [Überlegungen zu und Einschränkungen für Lebenszyklus-Hooks](#)
- [Zugehörige Ressourcen](#)
- [So funktionieren Lifecycle-Hooks in Auto Scaling Scaling-Gruppen](#)
- [Vorbereiten des Hinzufügens eines Lebenszyklus-Hook zu einer Auto-Scaling-Gruppe](#)

- [Steuern Sie die Aufbewahrung von Instances mit Richtlinien für den Instance-Lebenszyklus](#)
- [Abrufen des Ziellebenszyklus-Status durch Instance-Metadaten](#)
- [Fügen Sie Lifecycle-Hooks zu Ihrer Auto Scaling Scaling-Gruppe hinzu](#)
- [Eine Lebenszyklusaktion in einer Auto Scaling Scaling-Gruppe abschließen](#)
- [Tutorial: Verwenden Sie Datenskript- und Instance-Metadaten, um den Lebenszyklusstatus abzurufen](#)
- [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#)

Verfügbarkeit von Lebenszyklus-Hooks

In der folgenden Tabelle finden Sie die Lebenszyklus-Hooks, die für verschiedene Szenarien verfügbar sind

Veranstaltung	Instance-Start oder -Beendigung ¹	Maximale Instance-Lebensdauer : Ersatz-Instances	Instance-Aktualisierung : Ersatz-Instances	Kapazitätsausgleich : Ersatz-Instances	Warm-Pool : Instances, die den Warm-Pool betreten und verlassen
Startende Instance	✓	✓	✓	✓	✓
Endende Instance	✓	✓	✓	✓	✓

¹ Gilt für alle Starts und Beendigungen, unabhängig davon, ob sie automatisch oder manuell eingeleitet werden, z. B. wenn Sie die Optionen `SetDesiredCapacity` oder `TerminateInstanceInAutoScalingGroup` aufrufen. Gilt nicht, wenn Sie Instances zuordnen oder trennen, Instances in den Standby-Modus verschieben oder die Gruppe mit der Option „Löschen erzwingen“ löschen.

Überlegungen zu und Einschränkungen für Lebenszyklus-Hooks

Bei der Arbeit mit Lebenszyklus-Hooks sind die folgenden Hinweise und Einschränkungen zu beachten:

- Amazon EC2 Auto Scaling bietet einen eigenen Lebenszyklus, der die Verwaltung von Auto-Scaling-Gruppen unterstützt. Dieser Lebenszyklus unterscheidet sich von dem anderer EC2-Instances. Weitere Informationen finden Sie unter [Instance-Lebenszyklus bei Amazon EC2 Auto Scaling](#). Instances in einem Warm Pool haben auch einen eigenen Lebenszyklus, wie unter [Lebenszyklusstatusübergänge für Instances in einem Warm Pool](#) beschrieben.
- Standardmäßig werden Termination-Lifecycle-Hooks nach bestem Wissen und Gewissen ausgeführt. Wenn bei einem Terminierungs-Lifecycle-Hook eine Zeitüberschreitung eintritt oder abgebrochen wird, beendet Amazon EC2 Auto Scaling die Instance sofort. Sie können die Termination Lifecycle-Hooks mit einer Instance-Lifecycle-Richtlinie für die Aufbewahrung von Instances kombinieren. Weitere Informationen finden Sie unter [Steuern Sie die Aufbewahrung von Instances mit Richtlinien für den Instance-Lebenszyklus](#).
- Sie können Lebenszyklus-Hooks mit Spot-Instances verwenden, aber ein Lebenszyklus-Hook kann nicht verhindern, dass eine Instance beendet wird, wenn keine Kapazität mehr verfügbar ist, was jederzeit innerhalb eines zweiminütigen Unterbrechungshinweises passieren kann. Weitere Informationen finden Sie unter [Spot-Instance-Unterbrechungen](#) im Amazon EC2 EC2-Benutzerhandbuch. Sie können jedoch den Kapazitätsausgleich aktivieren, um Spot-Instances proaktiv zu ersetzen, die eine Neuausgleichsempfehlung vom Amazon-EC2-Spot-Service erhalten haben, ein Signal, das gesendet wird, wenn eine Spot-Instance einem erhöhten Unterbrechungsrisiko ausgesetzt ist. Weitere Informationen finden Sie unter [Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances](#).
- Instances können eine begrenzte Zeit lang in einem Wartestatus verbleiben. Das Standard-Timeout für einen Lebenszyklus-Hook beträgt eine Stunde (Heartbeat-Timeout). Es gibt auch ein globales Timeout, das die maximale Zeitspanne angibt, für die Sie eine Instance in einem Wartestatus belassen können. Das globale Timeout beträgt 48 Stunden bzw. das 100-fache der Heartbeat-Zeitüberschreitung, je nachdem, welcher Wert kleiner ist.
- Das Ergebnis des Lebenszyklus-Hooks kann entweder Abbrechen oder Fortsetzen sein. Wenn eine Instance gestartet wird, zeigt „Continue“ (Fortfahren) an, dass Ihre Aktionen erfolgreich waren und dass Amazon EC2 Auto Scaling die Instance in Betrieb nehmen kann. Andernfalls zeigt „Abandon“ (Abbruch) an, dass die benutzerdefinierten Aktionen fehlgeschlagen sind, und wir die Instance beenden und diese ersetzen können. Wenn die Instance beendet wird, erlauben sowohl „Abbrechen“ als auch „Fortfahren“ das Beenden der Instance. Allerdings stoppt „Abbrechen“ alle verbleibenden Aktionen, z. B. andere Lebenszyklus-Hooks, und „Fortfahren“ ermöglicht das Abschließen anderer Lebenszyklus-Hooks.

- Amazon EC2 Auto Scaling begrenzt die Rate, mit der Instances gestartet werden können, wenn die Lebenszyklus-Hooks konsistent fehlschlagen. Testen und Beheben Sie daher alle dauerhaften Fehler in Ihren Lebenszyklus-Aktionen.
- Das Erstellen und Aktualisieren von Lifecycle-Hooks mit dem AWS CLI CloudFormation, oder einem SDK bietet Optionen, die beim Erstellen eines Lifecycle-Hooks aus dem nicht verfügbar sind. AWS-Managementkonsole Beispielsweise wird das Feld zur Angabe des ARN eines SNS-Themas oder einer SQS-Warteschlange nicht in der Konsole angezeigt, da Amazon EC2 Auto Scaling bereits Ereignisse an Amazon sendet. EventBridge Diese Ereignisse können gefiltert und nach Bedarf an AWS Dienste wie Lambda, Amazon SNS und Amazon SQS umgeleitet werden.
- Sie können einer Auto Scaling Scaling-Gruppe während der Erstellung mehrere Lifecycle-Hooks hinzufügen, indem Sie die [CreateAutoScalingGroup](#)API mit dem AWS CLI CloudFormation, oder einem SDK aufrufen. Jeder Hook muss jedoch das gleiche Benachrichtigungsziel und die gleiche IAM-Rolle haben, falls angegeben. Um Lifecycle-Hooks mit unterschiedlichen Benachrichtigungszielen und unterschiedlichen Rollen zu erstellen, erstellen Sie die Lifecycle-Hooks nacheinander in separaten Aufrufen der [PutLifecycleHook](#)API.
- Wenn Sie zum Beispiel einen Lebenszyklus-Hook für den Instance-Start hinzufügen, beginnt die Übergangsfrist für die Zustandsprüfung, sobald die Instance den Status InService erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartefrist für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).

Überlegungen zur Skalierung

- Dynamische Skalierungsrichtlinien werden als Reaktion auf CloudWatch metrische Daten wie CPU und Netzwerk-I/O, die über mehrere Instanzen hinweg aggregiert werden, nach innen und außen skaliert. Beim Skalieren zählt Amazon EC2 Auto Scaling eine neue Instance nicht sofort zu den aggregierten Instance-Metriken der Auto-Scaling-Gruppe. Es wartet, bis die Instance den Status InService erreicht hat und der Instance-Warmup abgeschlossen ist. Weitere Informationen finden Sie unter [Leistungsaspekte der Skalierung](#) im Thema Aufwärmen der Standard-Instance.
- Beim Skalieren spiegeln die aggregierten Instance-Metriken möglicherweise nicht sofort die Entfernung einer beendeten Instance wider. Die terminierende Instance wird nicht mehr zu den aggregierten Instance-Metriken der Gruppe gezählt, kurz nachdem der Amazon EC2 Auto Scaling Terminierungsworkflow beginnt.
- In den meisten Fällen, in denen Lebenszyklus-Hooks aufgerufen werden, werden die Skalierungsaktivitäten aufgrund einfacher Skalierungsrichtlinien angehalten, bis die Lifecycle-Aktionen abgeschlossen sind und die Ruhephase abgelaufen ist. Bei einem Festlegen eines

langen Intervalls für die Ruhephase dauert es länger, bis die Skalierung fortgesetzt werden kann. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks können zusätzliche Verzögerungen verursachen](#) im Thema Ruhephase. Im Allgemeinen empfehlen wir, keine einfachen Skalierungsrichtlinien zu verwenden, wenn Sie stattdessen entweder eine Stufenskalierung oder eine Zielverfolgungsskalierungs-Richtlinie verwenden können.

Zugehörige Ressourcen

Ein Einführungsvideo finden Sie unter [AWS re:Invent 2018: Capacity Management Made Easy with Amazon EC2 Auto Scaling](#) on YouTube

Wir stellen einige JSON- und YAML-Vorlagenausschnitte zur Verfügung, anhand derer Sie verstehen können, wie Sie Lifecycle-Hooks in Ihren Stack-Vorlagen deklarieren. CloudFormation Weitere Informationen finden Sie in der [AWS::AutoScaling::LifecycleHook](#) Referenz im AWS CloudFormation Benutzerhandbuch.

Sie können auch unser [GitHubRepository](#) besuchen, um Beispielvorlagen und Benutzerdatenskripte für Lifecycle-Hooks herunterzuladen.

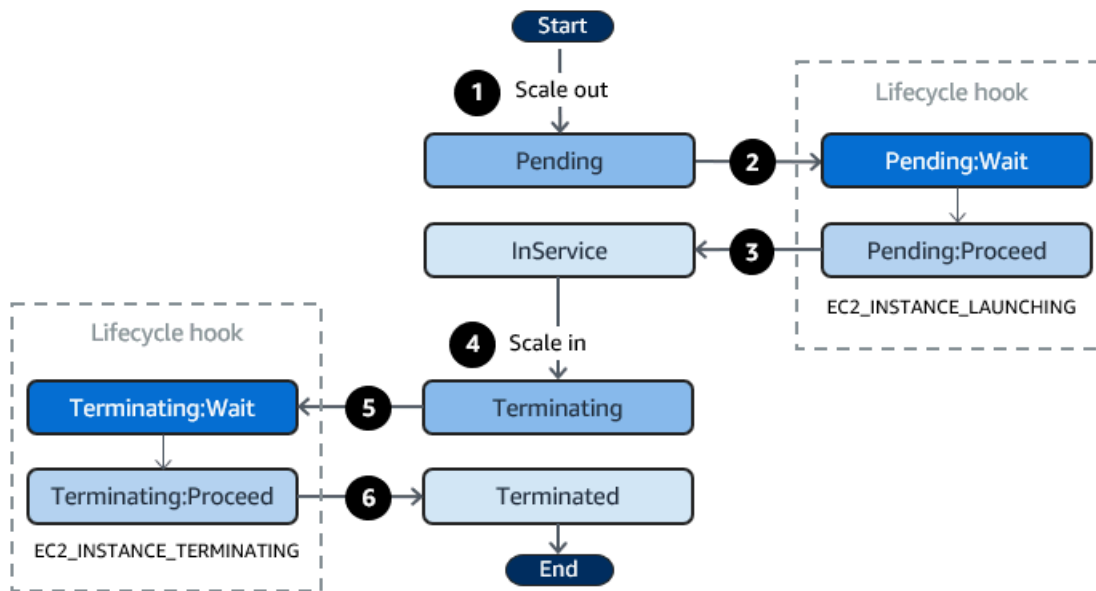
Beispiele für die Verwendung von Lebenszyklus-Hooks finden Sie in den folgenden Blog-Beiträgen.

- [Aufbau eines Backup-Systems für skalierte Instances mithilfe von Lambda und Amazon EC2 Run Command](#)
- [Code vor dem Beenden einer EC2 Auto Scaling-Instance ausführen.](#)

So funktionieren Lifecycle-Hooks in Auto Scaling Scaling-Gruppen

Eine Amazon-EC2-Instance wechselt von dem Zeitpunkt, an dem sie gestartet wird, bis zu ihrer Beendigung durch verschiedene Status. Sie können benutzerdefinierte Aktionen für Ihre Auto-Scaling-Gruppe erstellen, die ausgeführt werden, wenn eine Instance aufgrund eines Lebenszyklus-Hooks in einen Wartezustand übergeht.

Die folgende Abbildung zeigt die Übergänge zwischen Auto Scaling Scaling-Instanzzuständen, wenn Sie Lifecycle-Hooks für Scale-Out und Scale-In verwenden.



Wie im obigen Diagramm gezeigt:

1. Die Auto-Scaling-Gruppe reagiert auf ein horizontales Skalierungsereignis und beginnt mit dem Starten einer Instance.
2. Der Lebenszyklus-Hook versetzt die Instance in einen Wartestatus (Pending:Wait) und führt dann eine benutzerdefinierte Aktion aus.

Die Instance bleibt in einem Wartezustand, bis Sie entweder die Lebenszyklusaktion beenden oder der Timeout-Zeitraum endet. Standardmäßig verbleibt die Instance eine Stunde lang im Wartestatus, dann führt die Auto-Scaling-Gruppe den Start der Instance fort (Pending:Proceed). Wird mehr Zeit benötigt, können Sie die Zeit bis zur Zeitüberschreitung zurücksetzen, indem Sie eine Heartbeat-Benachrichtigung aufzeichnen. Wenn Sie die Lebenszyklusaktion bei Vollendung der benutzerdefinierten Aktion und bevor der Zeitüberschreitungszeitraums abgelaufen ist, abschließen, endet der Zeitraum und die Auto-Scaling-Gruppe setzt den Startvorgang fort.

3. Die Instance tritt in den InService-Zustand und die Kulanfrist der Zustandsprüfung beginnt. Bevor die Instance jedoch den InService-Status erreicht, wird sie, wenn die Auto-Scaling-Gruppe einem Elastic Load Balancing zugeordnet ist, beim Load Balancer registriert, und der Load Balancer beginnt mit der Überprüfung seines Zustands. Nach Ablauf der Kulanfrist für die Zustandsprüfung beginnt Amazon EC2 Auto Scaling, den Zustand der Instance zu prüfen.
4. Die Auto-Scaling-Gruppe reagiert auf ein horizontales Skalierungsereignis und beginnt mit dem Beenden einer Instance. Wenn die Auto-Scaling-Gruppe mit Elastic Load Balancing verwendet wird, wird die beendende Instance zunächst beim Load Balancer registriert. Wenn Connection Draining für den Load Balancer aktiviert ist, akzeptiert die Instance keine neuen Verbindungen

und wartet darauf, dass vorhandene Verbindungen abgebaut werden, bevor die Registrierung abgeschlossen wird.

5. Der Lebenszyklus-Hook versetzt die Instance in einen Wartestatus (`Terminating:Wait`) und führt dann eine benutzerdefinierte Aktion aus.

Die Instance bleibt in einem Wartezustand, bis Sie entweder die Lebenszyklusaktion beenden oder der Timeout-Zeitraum endet (standardmäßig eine Stunde). Nachdem Sie den Lebenszyklus-Hook abgeschlossen haben oder der Timeout-Zeitraum abläuft, wechselt die Instance in den nächsten Status (`Terminating:Proceed`).

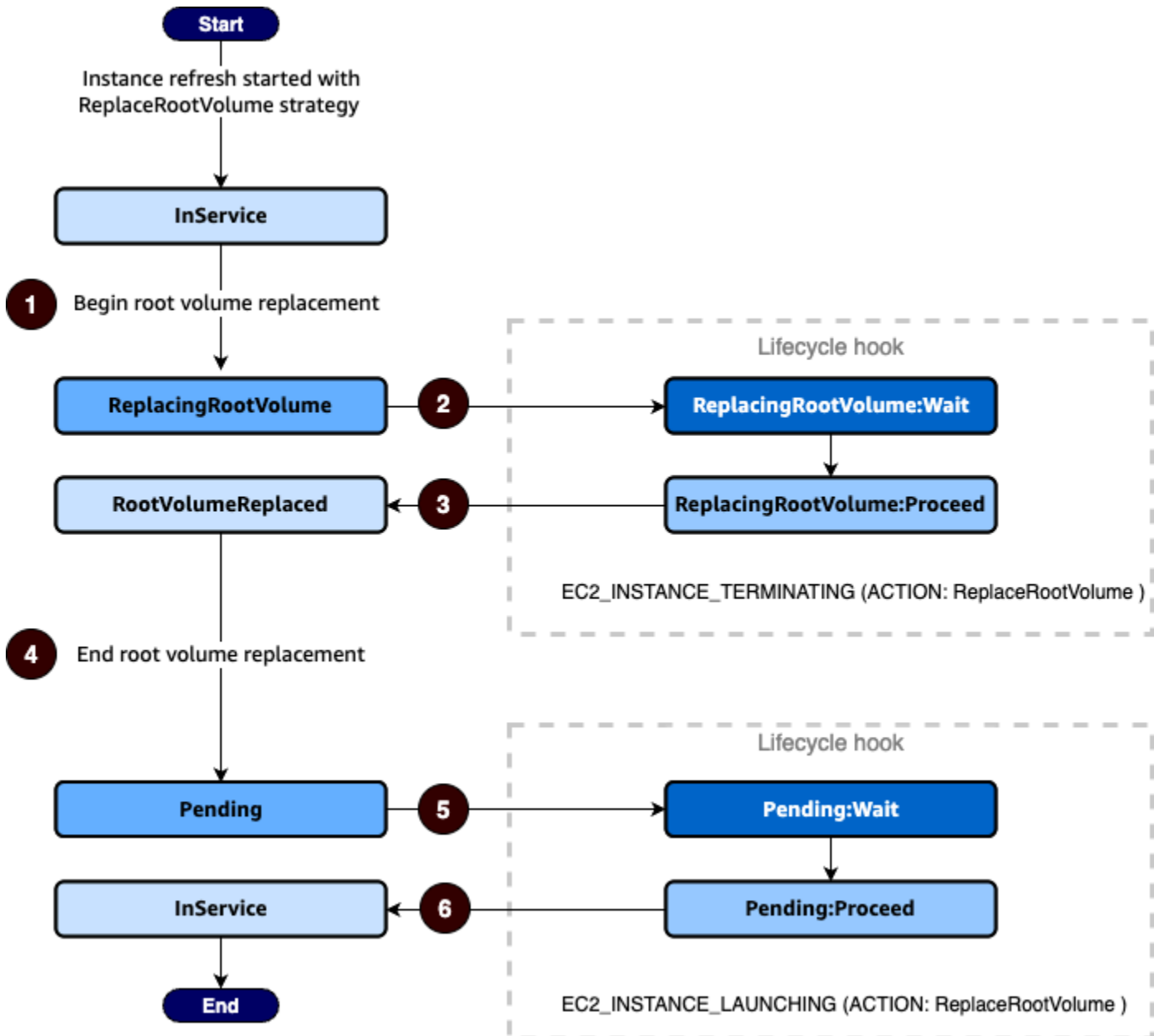
6. Die Instance wurde beendet.

⚠ Important

Instances in einem Warm Pool haben auch einen eigenen Lebenszyklus mit entsprechenden Wartestatus, wie unter [Lebenszyklusstatusübergänge für Instances in einem Warm Pool](#) beschrieben.

Statusübergänge im Lebenszyklus von Instances, bei denen das Root-Volume ausgetauscht wird

Das folgende Diagramm zeigt den Übergang zwischen Auto Scaling Scaling-Instanzzuständen, wenn Sie Lifecycle-Hooks zum Ersetzen des Root-Volumes verwenden:



Wie im obigen Diagramm gezeigt:

1. Die Auto Scaling Scaling-Gruppe reagiert auf eine Instanzaktualisierung und wählt eine Instance für den Austausch des Root-Volumes aus. Die Instance wechselt in den `ReplacingRootVolume` Status. Wenn die Instance bei einem Load Balancer registriert ist, wird sie vom Load Balancer abgemeldet.
2. Der Lebenszyklus-Hook versetzt die Instance in einen Wartestatus (`ReplacingRootVolume:Wait`) und führt dann eine benutzerdefinierte Aktion aus. Die Instance bleibt in einem Wartezustand, bis Sie entweder die Lebenszyklusaktion beenden

oder der Timeout-Zeitraum endet. Wenn Sie die Lebenszyklusaktion abschließen, wenn die benutzerdefinierte Aktion abgeschlossen ist und der Timeout-Zeitraum noch nicht abgelaufen ist, endet der Zeitraum und die Auto Scaling Scaling-Gruppe setzt den Austausch des Stammvolumens fort.

3. Die Instance schließt den Austausch des Root-Volumens ab und wechselt in den `RootVolumeReplaced` Status.
4. Die Instance wechselt in den `Pending` Status.
5. Der Lebenszyklus-Hook versetzt die Instance in einen Wartestatus (`Pending:Wait`) und führt dann eine benutzerdefinierte Aktion aus. Die Instance verbleibt im Wartestatus, entweder bis Sie die Lebenszyklusaktion abgeschlossen haben oder bis der Timeout-Zeitraum abgelaufen ist. Nachdem Sie den Lebenszyklus-Hook abgeschlossen haben oder der Timeout-Zeitraum abläuft, wechselt die Instance in den nächsten Status (`Pending:Proceed`).
6. Die Instanz wechselt in den `InService` Status. Wenn die Auto Scaling Scaling-Gruppe jedoch einem Elastic Load Balancing Load Balancer zugeordnet ist, bevor die Instance den `InService` Status erreicht, wird die Instance beim Load Balancer registriert.

Vorbereiten des Hinzufügens eines Lebenszyklus-Hook zu einer Auto-Scaling-Gruppe

Stellen Sie sicher, dass Ihr Benutzerdatenskript oder Ihr Benachrichtigungsziel korrekt eingerichtet ist, bevor Sie Ihrer Auto-Scaling-Gruppe einen Lebenszyklus-Hook hinzufügen.

- Um ein Benutzerdatenskript zu nutzen, um benutzerdefinierte Aktionen für Ihre Instances während des Starts auszuführen, müssen Sie kein Benachrichtigungsziel konfigurieren. Sie müssen jedoch bereits die Startvorlage oder die Startkonfiguration erstellt haben, die Ihr Benutzerdatenskript angibt und es Ihrer Auto-Scaling-Gruppe zuordnet. Weitere Informationen zu Benutzerdatenskripten finden Sie unter [Befehle auf Ihrer Linux-Instance beim Start ausführen](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Um Amazon EC2 Auto Scaling zu signalisieren, wenn die Lifecycle-Aktion abgeschlossen ist, müssen Sie den [CompleteLifecycleAction](#) API-Aufruf zum Skript hinzufügen und manuell eine IAM-Rolle mit einer Richtlinie erstellen, die es Auto Scaling-Instances ermöglicht, diese API aufzurufen. Ihre Startvorlage oder Startkonfiguration muss diese Rolle mithilfe eines IAM-Instance-Profiles angeben, das beim Start an Ihre Amazon-EC2-Instances angehängt wird. Weitere Informationen erhalten Sie unter [Eine Lebenszyklusaktion in einer Auto Scaling Scaling-Gruppe abschließen](#) und [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#).

- Damit Lambda Amazon EC2 Auto Scaling signalisieren kann, wenn die Lebenszyklusaktion abgeschlossen ist, müssen Sie den [CompleteLifecycleAction](#) API-Aufruf zum Funktionscode hinzufügen. Sie müssen auch eine IAM-Richtlinie an die Ausführungsrolle der Funktion angehängt haben, um Lambda die Berechtigung zum Vervollständigen von Lebenszyklus-Aktionen zu erteilen. Weitere Informationen finden Sie unter [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#).
- Um einen Service wie Amazon SNS oder Amazon SQS zum Ausführen einer benutzerdefinierten Aktion verwenden zu können, müssen Sie bereits das SNS-Thema oder die SQS-Warteschlange erstellt haben und über den Amazon-Ressourcennamen (ARN) verfügen. Sie müssen auch bereits die IAM-Rolle erstellt haben, die Amazon EC2 Auto Scaling Zugriff auf Ihr SNS-Thema oder SQS-Ziel ermöglicht, und über deren ARN verfügen. Weitere Informationen finden Sie unter [Konfigurieren eines Benachrichtigungsziels für Lebenszyklus-Benachrichtigungen](#).

Note

Wenn Sie einen Lifecycle-Hook in der Konsole hinzufügen, sendet Amazon EC2 Auto Scaling standardmäßig Lebenszyklusereignisbenachrichtigungen an Amazon EventBridge. Die Verwendung EventBridge eines Benutzerdatenskripts ist eine empfohlene bewährte Methode. Um einen Lifecycle-Hook zu erstellen, der Benachrichtigungen direkt an Amazon SNS, Amazon SQS oder sendet, verwenden Sie das AWS Lambda, oder ein SDK AWS CLI AWS CloudFormation, um den Lifecycle-Hook hinzuzufügen.

Konfigurieren eines Benachrichtigungsziels für Lebenszyklus-Benachrichtigungen

Sie können einer Auto-Scaling-Gruppe Lebenszyklus-Hooks hinzufügen, um benutzerdefinierte Aktionen auszuführen, wenn eine Instance in einen Wartestatus wechselt. Sie können einen Zielservice auswählen, der diese Aktionen abhängig von Ihrem bevorzugten Entwicklungsansatz ausführt.

Es gibt vier verschiedene Ansätze für die Implementierung von Benachrichtigungszielen für Lifecycle-Hooks:

- Amazon EventBridge — Erhalten Sie die Benachrichtigungen und führen Sie die gewünschten Aktionen aus.
- Amazon Simple Notification Service (Amazon SNS) — Erstellen Sie ein Thema für die Veröffentlichung von Benachrichtigungen. Kunden können das SNS-Thema abonnieren und veröffentlichte Nachrichten über ein unterstütztes Protokoll empfangen.

- Amazon Simple Queue Service (Amazon SQS) — Tauschen Sie Nachrichten über ein Abfragemodell aus.
- AWS Lambda— Rufen Sie eine Lambda-Funktion auf, die die gewünschte Aktion ausführt.

Als bewährte Methode empfehlen wir die Verwendung von EventBridge. Die an Amazon SNS und Amazon SQS gesendeten Benachrichtigungen enthalten dieselben Informationen wie die Benachrichtigungen, an die Amazon EC2 Auto Scaling sendet. EventBridge Bisher bestand die Standardpraxis darin EventBridge, eine Benachrichtigung an SNS oder SQS zu senden und einen anderen Service in SNS oder SQS zu integrieren, um programmatische Aktionen durchzuführen. Heute stehen EventBridge Ihnen mehr Optionen zur Verfügung, auf welche Dienste Sie abzielen können, und erleichtert die Verarbeitung von Ereignissen mithilfe einer serverlosen Architektur.

Denken Sie daran: Wenn Sie über ein Benutzerdatenskript in Ihrer Startvorlage- oder Startkonfiguration verfügen, das Ihre Instances beim Starten konfiguriert, müssen Sie keine Benachrichtigungen erhalten, um benutzerdefinierte Aktionen für Ihre Instances auszuführen.

In den folgenden Verfahren wird beschrieben, wie Sie Ihr Benachrichtigungsziel einrichten.

Inhalt

- [Benachrichtigungen an Lambda weiterleiten mit EventBridge](#)
- [Benachrichtigungen über Amazon SNS erhalten](#)
- [Benachrichtigungen über Amazon SQS erhalten](#)
- [Benachrichtigungen AWS Lambda direkt weiterleiten](#)
- [Beispiel für Benachrichtigungsnachricht](#)

Important

Die EventBridge Regel, die Lambda-Funktion, das Amazon SNS SNS-Thema und die Amazon SQS SQS-Warteschlange, die Sie mit Lifecycle-Hooks verwenden, müssen sich immer in derselben Region befinden, in der Sie Ihre Auto Scaling Scaling-Gruppe erstellt haben.

Benachrichtigungen an Lambda weiterleiten mit EventBridge

Sie können eine EventBridge Regel so konfigurieren, dass sie eine Lambda-Funktion aufruft, wenn eine Instanz in den Wartezustand wechselt. Amazon EC2 Auto Scaling sendet eine Benachrichtigung EventBridge über ein Lifecycle-Ereignis an die Instance, die gestartet oder beendet wird, sowie ein Token, mit dem Sie die Lifecycle-Aktion steuern können. Beispiele für diese Ereignisse finden Sie unter [Ereignis-Referenz für Amazon EC2 Auto Scaling](#).

Note

Wenn Sie die verwenden, AWS-Managementkonsole um eine Ereignisregel zu erstellen, fügt die Konsole automatisch die IAM-Berechtigungen hinzu, die erforderlich sind, um die EventBridge Berechtigung zum Aufrufen Ihrer Lambda-Funktion zu erteilen. Wenn Sie eine Ereignisregel mit AWS CLI erstellen, müssen Sie diese Berechtigung ausdrücklich erteilen. Informationen zum Erstellen von Ereignisregeln in der EventBridge Konsole finden Sie im [EventBridge Amazon-Benutzerhandbuch unter Erstellen von EventBridge Amazon-Regeln, die auf Ereignisse reagieren](#).

– oder –

Ein einführendes Tutorial, das sich an Konsolenbenutzer richtet, finden Sie unter [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#). Dieses Tutorial zeigt Ihnen, wie Sie eine einfache Lambda-Funktion erstellen, die auf Startereignisse wartet und diese in ein CloudWatch Logs-Protokoll schreibt.

Um eine EventBridge Regel zu erstellen, die eine Lambda-Funktion aufruft

1. Erstellen Sie mithilfe der [Lambda-Konsole](#) eine Lambda-Funktion und notieren Sie ihren Amazon-Ressourcennamen (ARN). Beispiel, `arn:aws:lambda:region:123456789012:function:my-function`. Sie benötigen den ARN, um ein EventBridge Ziel zu erstellen. Weitere Informationen finden Sie unter [Erste Schritte mit Lambda](#) im AWS Lambda -Entwicklerhandbuch.
2. Um eine Regel zu erstellen, die auf Ereignisse für den Start der Instance passt, verwenden Sie den folgenden [put-rule](#)-Befehl.

```
aws events put-rule --name my-rule --event-pattern file://pattern.json --state  
ENABLED
```

Im folgenden Beispiel wird die Aktion `pattern.json` für eine Instance zum Starten des Lebenszyklus veranschaulicht. Ersetzen Sie den Text *italics* durch den Namen Ihrer Auto Scaling Scaling-Gruppe.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "AutoScalingGroupName": [ "my-asg" ]
  }
}
```

Wenn der Befehl erfolgreich ausgeführt wird, EventBridge antwortet er mit dem ARN der Regel. Notieren Sie diesen ARN. Sie müssen ihn in Schritt 4 eingeben.

Um eine Regel zu erstellen, die mit anderen Ereignissen übereinstimmt, ändern Sie das Ereignismuster. Weitere Informationen finden Sie unter [Wird EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen verwendet](#).

3. Verwenden Sie Folgendes, um die Lambda-Funktion anzugeben, die als Ziel für die Regel verwendet werden soll: [put-targets](#)-Befehl.

```
aws events put-targets --rule my-rule --targets
  Id=1,Arn=arn:aws:lambda:region:123456789012:function:my-function
```

Im vorherigen Befehl *my-rule* ist dies der Name, den Sie in Schritt 2 für die Regel angegeben haben, und der Wert für den `Arn` Parameter ist der ARN der Funktion, die Sie in Schritt 1 erstellt haben.

4. Um Berechtigungen hinzuzufügen, die es der Regel erlauben, Ihre Lambda-Funktion aufzurufen, verwenden Sie den folgenden Lambda [add-permission](#)-Befehl. Dieser Befehl vertraut dem EventBridge Dienstprinzipal (`events.amazonaws.com`) und beschränkt die Berechtigungen auf die angegebene Regel.

```
aws lambda add-permission --function-name my-function --statement-id my-unique-id \
  --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
  arn:aws:events:region:123456789012:rule/my-rule
```

Beim vorhergehenden Befehl:

- *my-function* ist der Name der Lambda-Funktion, die die Regel als Ziel verwenden soll.
- *my-unique-id* ist ein eindeutiger Bezeichner, den Sie definieren, um die Anweisung in der Lambda-Funktionsrichtlinie zu beschreiben.
- *source-arn* ist der ARN der EventBridge Regel.

Wird der Befehl erfolgreich ausgeführt, erhalten Sie eine Ausgabe ähnlich der folgenden:

```
{
  "Statement": "{\"Sid\":\"my-unique-id\",
    \"Effect\":\"Allow\",
    \"Principal\":{\"Service\":\"events.amazonaws.com\"},
    \"Action\":\"lambda:InvokeFunction\",
    \"Resource\":\"arn:aws:lambda:us-west-2:123456789012:function:my-function\",
    \"Condition\":
      {\"ArnLike\":
        {\"AWS:SourceArn\":
          \"arn:aws:events:us-west-2:123456789012:rule/my-rule\"}}}"
}
```

Der Statement-Wert ist eine JSON-Zeichenfolgenversion der Anweisung, die der Lambda-Funktionsrichtlinie hinzugefügt wurde.

5. Nachdem Sie diese Anweisungen befolgt haben, fahren Sie mit [Fügen Sie Lifecycle-Hooks zu Ihrer Auto Scaling Scaling-Gruppe hinzu](#) fort.

Benachrichtigungen über Amazon SNS erhalten

Sie können Amazon SNS dazu verwenden, ein Benachrichtigungsziel (ein SNS-Thema) für den Empfang von Nachrichten im Falle einer Lebenszyklusaktion einzurichten. Amazon SNS sendet die Benachrichtigungen dann an die abonnierten Empfänger. Solange das Abonnement nicht bestätigt ist, werden keine Benachrichtigungen, die zum Thema veröffentlicht wurden, an die Empfänger gesendet.

Einrichten von Benachrichtigungen mithilfe von Amazon SNS

1. Erstellen Sie ein Amazon SNS-Thema mithilfe der [Amazon SNS Konsole](#) oder dem folgenden [create-topic](#)-Befehl. Stellen Sie sicher, dass sich das Thema in derselben Region befindet wie

die verwendete Auto-Scaling-Gruppe. Weitere Informationen finden Sie unter [Erste Schritte mit Amazon SNS](#) im Benutzerhandbuch für Amazon Simple Notification Service.

```
aws sns create-topic --name my-sns-topic
```

2. Notieren Sie den Amazon-Ressourcennamen (ARN) des Themas, zum Beispiel `arn:aws:sns:region:123456789012:my-sns-topic`. Sie benötigen ihn, um den Lebenszyklus-Hook zu erstellen.
3. Erstellen Sie eine IAM-Servicerolle, um Amazon EC2 Auto Scaling Zugriff auf Ihr Amazon SNS-Benachrichtigungsziel zu gewähren.

So gewähren Sie Amazon EC2 Auto Scaling Zugriff auf Ihr SNS-Thema

- a. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
 - b. Wählen Sie im Navigationsbereich auf der linken Seite Roles (Rollen).
 - c. Wählen Sie Rolle erstellen aus.
 - d. Wählen Sie für Select trusted entity (Vertrauenswürdige Entität auswählen) die Option AWS -Dienst.
 - e. Wählen Sie für Ihren Anwendungsfall unter Use cases for other AWS services (Anwendungsfälle für andere -Dienste), EC2 Auto Scaling (EC2 Auto Scaling) und dann EC2 Auto Scaling Notification Access (Zugriff auf EC2-Auto-Scaling-Benachrichtigungen) aus.
 - f. Klicken Sie zweimal auf Next (Weiter), um zur Seite Name, review, and create (Benennen, überprüfen und erstellen) zu gelangen.
 - g. Geben Sie für Role Name (Name der Rolle) einen Namen für Ihre Rolle ein (z. B. **my-notification-role**) und wählen Sie dann Create role (Rolle erstellen).
 - h. Wählen Sie auf der Seite Roles (Rollen) die gerade erstellte Rolle aus, um die Seite Summary (Übersicht) zu öffnen. Notieren Sie sich den ARN der Rolle. Beispiel, `arn:aws:iam::123456789012:role/my-notification-role`. Sie benötigen ihn, um den Lebenszyklus-Hook zu erstellen.
4. Nachdem Sie diese Anweisungen befolgt haben, fahren Sie mit [Hinzufügen von Lebenszyklus-Hooks \(AWS CLI\)](#) fort.

Benachrichtigungen über Amazon SQS erhalten

Sie können Amazon SQS dazu verwenden, ein Benachrichtigungsziel für den Empfang von Nachrichten im Falle einer Lebenszyklusaktion einzurichten. Ein Warteschlangen-Verbraucher muss dann eine SQS-Warteschlange abfragen, um auf diese Benachrichtigungen zu reagieren.

Important

FIFO-Warteschlangen sind nicht kompatibel mit Lebenszyklus-Hooks.

Einrichten von Benachrichtigungen mithilfe von Amazon SQS

1. Mit der [Amazon SQS-Konsole](#) erstellen Sie eine SQS-Warteschlange. Stellen Sie sicher, dass sich die Warteschlange in derselben Region befindet wie die von Ihnen verwendete Auto-Scaling-Gruppe. Weitere Informationen finden Sie unter [Erste Schritte mit Amazon SQS](#) im Benutzerhandbuch für Amazon Simple Queue Service.
2. Notieren Sie den ARN der Warteschlange, z. B. `arn:aws:sqs:us-west-2:123456789012:my-sqs-queue`. Sie benötigen ihn, um den Lebenszyklus-Hook zu erstellen.
3. Erstellen Sie eine IAM-Servicerolle, um Amazon EC2 Auto Scaling Zugriff auf Ihr Amazon SQS-Benachrichtigungsziel zu gewähren.

So gewähren Sie Amazon EC2 Auto Scaling Zugriff auf Ihre SQS-Warteschlange

- a. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
- b. Wählen Sie im Navigationsbereich auf der linken Seite Roles (Rollen).
- c. Wählen Sie Rolle erstellen aus.
- d. Wählen Sie für Select trusted entity (Vertrauenswürdige Entität auswählen) die Option AWS-Dienst.
- e. Wählen Sie für Ihren Anwendungsfall unter Use cases for other AWS services (Anwendungsfälle für andere -Dienste), EC2 Auto Scaling (EC2 Auto Scaling) und dann EC2 Auto Scaling Notification Access (Zugriff auf EC2-Auto-Scaling-Benachrichtigungen) aus.
- f. Klicken Sie zweimal auf Next (Weiter), um zur Seite Name, review, and create (Benennen, überprüfen und erstellen) zu gelangen.
- g. Geben Sie für Role Name (Name der Rolle) einen Namen für Ihre Rolle ein (z. B. **my-notification-role**) und wählen Sie dann Create role (Rolle erstellen).

- h. Wählen Sie auf der Seite Roles (Rollen) die gerade erstellte Rolle aus, um die Seite Summary (Übersicht) zu öffnen. Notieren Sie sich den ARN der Rolle. Beispiel, `arn:aws:iam::123456789012:role/my-notification-role`. Sie benötigen ihn, um den Lebenszyklus-Hook zu erstellen.
4. Nachdem Sie diese Anweisungen befolgt haben, fahren Sie mit [Hinzufügen von Lebenszyklus-Hooks \(AWS CLI\)](#) fort.

Benachrichtigungen AWS Lambda direkt weiterleiten

Sie können eine Lambda-Funktion als Benachrichtigungsziel verwenden, wenn eine Lebenszyklusaktion stattfindet.

Um Benachrichtigungen direkt weiterzuleiten AWS Lambda

1. Öffnen Sie die Seite [Funktionen](#) der Lambda-Konsole.
2. Wählen Sie die gewünschte Lambda-Funktion aus.

Wenn Sie eine neue Lambda-Funktion erstellen möchten, finden Sie unter [So erstellen Sie die Lambda-Funktion:](#)

3. Wählen Sie die Registerkarte Konfiguration und dann Berechtigungen.
4. Scrollen Sie nach unten bis zu Ressourcenbasierte Richtlinie und wählen Sie dann Hinzufügen von Berechtigungen aus. Eine ressourcenbasierte Richtlinie wird verwendet, um dem Prinzipal, der in der Richtlinie angegeben ist, Berechtigungen zum Aufrufen Ihrer Funktion zu erteilen. In diesem Fall wird der Prinzipal die [serviceverknüpfte Amazon EC2 Auto Scaling-Rolle](#) sein, die der Auto-Scaling-Gruppe zugeordnet ist.
5. In der Richtlinienanweisung konfigurieren Sie Ihre Berechtigungen:
 - a. Wählen Sie AWS-Konto.
 - b. Für Prinzipal geben Sie den ARN der aufrufenden serviceverknüpften Rolle ein, z. B. **`arn:aws:iam::<aws-account-id>:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling`**.
 - c. Wählen Sie für Aktion die Option `lambda: InvokeFunction` aus.
 - d. Für Anweisungs-ID geben Sie eine eindeutige Anweisungs-ID ein, wie z. B. **`AllowInvokeByAutoScaling`**.
 - e. Wählen Sie Speichern.

6. Nachdem Sie diese Anweisungen befolgt haben, fahren Sie mit [Hinzufügen von Lebenszyklus-Hooks \(AWS CLI\)](#) fort.

Beispiel für Benachrichtigungsnachricht

Dieser Abschnitt enthält ein Beispiel für Benachrichtigungen für Amazon SNS, Amazon SQS und AWS Lambda

Während sich die Instance im Wartestatus befindet, wird eine Nachricht an Amazon SNS, Amazon SQS und das AWS Lambda Benachrichtigungsziel veröffentlicht.

Die Nachricht enthält die folgenden Informationen:

- **Origin**— Ein Ort, von dem eine EC2-Instance kommt.
- **Destination**— Ein Ort, an den eine EC2-Instance gehen wird.
- **LifecycleActionToken** – Das Token der Lebenszyklusaktion
- **AccountId**— Die AWS-Konto ID.
- **AutoScalingGroupName** – Der Name der Auto-Scaling-Gruppe.
- **LifecycleHookName** – Der Name des Lebenszyklus-Hooks.
- **EC2InstanceId** – Die ID der EC2-Instance.
- **LifecycleTransition** – Die Art des Lebenszyklus-Hooks.
- **NotificationMetadata** – Die Benachrichtigungsmetadaten.

Im Folgenden finden Sie ein Beispiel für eine Benachrichtigungsmeldung.

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:36:26.533Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Origin: EC2
Destination: AutoScalingGroup
LifecycleActionToken: 71514b9d-6a40-4b26-8523-05e7ee35fa40
AccountId: 123456789012
AutoScalingGroupName: my-asg
LifecycleHookName: my-hook
EC2InstanceId: i-0598c7d356eba48d7
LifecycleTransition: autoscaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata: hook message metadata
```

Beispiel für Benachrichtigungsnachricht testen

Wenn Sie zum ersten Mal einen Lebenszyklus-Hook hinzufügen, wird eine Testbenachrichtigung für das Benachrichtigungsziel veröffentlicht. Im Folgenden finden Sie ein Beispiel für eine Testbenachrichtigungsnachricht.

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:35:52.359Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Event: autoscaling:TEST_NOTIFICATION
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg
```

Note

Beispiele für Ereignisse, die von Amazon EC2 Auto Scaling an übermittlemt wurden EventBridge, finden Sie unter [Ereignis-Referenz für Amazon EC2 Auto Scaling](#).

Steuern Sie die Aufbewahrung von Instances mit Richtlinien für den Instance-Lebenszyklus

Richtlinien für den Instance-Lebenszyklus bieten Schutz vor der Kündigung von Amazon EC2 Auto Scaling, wenn eine Aktion zum Beenden des Lebenszyklus abgebrochen wird. Im Gegensatz zu Lifecycle-Hooks allein sollen Richtlinien für den Instance-Lebenszyklus sicherstellen, dass Instances in einen beibehaltenen Zustand übergehen, wenn das ordnungsgemäße Herunterfahren nicht erfolgreich abgeschlossen werden kann.

Wann sollten Instance-Lifecycle-Richtlinien verwendet werden

Verwenden Sie Richtlinien für den Instanzlebenszyklus, wenn ein ordnungsgemäßes Herunterfahren Ihrer Anwendung nicht optional, sondern obligatorisch ist und ein fehlgeschlagenes Herunterfahren ein manuelles Eingreifen erfordert. Häufige Anwendungsfälle umfassen:

- Stateful-Anwendungen, die vor der Kündigung die Datenpersistenz abschließen müssen.

- Anwendungen, die längere Entladezeiten erfordern, die das maximale Lifecycle-Hook-Timeout von 48 Stunden überschreiten können.
- Workloads, die sensible Daten verarbeiten und bei denen die Bereinigung fehlschlägt oder unvollständig ist, können zu Datenverlust oder -beschädigung führen.
- Geschäftskritische Dienste, bei denen ein abruptes Herunterfahren die Verfügbarkeit beeinträchtigt.

Weitere Informationen zur ordnungsgemäßen Handhabung der Instanzbeendigung finden Sie unter [Gestalten Sie Ihre Anwendungen so, dass sie die Instance-Kündigung ordnungsgemäß handhaben](#)

So funktionieren Instance-Lebenszyklus-Richtlinien mit Terminierungs-Lifecycle-Hooks

Richtlinien für den Instanzlebenszyklus funktionieren in Kombination mit Terminierungs-Lifecycle-Hooks, nicht als Ersatz. Der Prozess besteht aus mehreren Phasen:

1. Die Aktionen im Lebenszyklus der Kündigung werden ausgeführt. Wenn Amazon EC2 Auto Scaling eine Instance zur Kündigung auswählt, werden Ihre Terminierungs-Lifecycle-Hooks aufgerufen und die Instance wechselt in den `Terminating:Wait` Status, in dem sie mit der Ausführung der Terminierungslebenszyklus-Aktionen beginnen kann.
2. Der Versuch, das System ordnungsgemäß herunterzufahren, beginnt. Ihre Anwendung, die entweder auf der Instance oder über eine Steuerungsebene ausgeführt wird, erhält die Benachrichtigung über die Aktion zum Beenden des Lebenszyklus und beginnt mit Verfahren zum ordnungsgemäßen Herunterfahren, wie z. B. das Löschen von Verbindungen, das Abschließen laufender Arbeiten oder das Übertragen von Daten.
3. Die Aktionen im Lebenszyklus der Kündigung sind abgeschlossen. Eine Aktion im Lebenszyklus einer Kündigung kann mit `CONTINUE` oder mit einem `ABANDON` Ergebnis abgeschlossen werden.
4. Die Richtlinie für den Instanzlebenszyklus bewertet die Situation. Wenn keine Instance-Lebenszyklus-Richtlinie konfiguriert ist, wird die Instance sofort beendet, auch wenn die Aktion zum Beenden des Lebenszyklus erfolgreich abgeschlossen wurde `ABANDON`. Wenn eine Instance-Lebenszyklus-Richtlinie so konfiguriert ist, dass Instances aktiviert bleiben `TerminateHookAbandon`, wechselt die Instance in den Status `Beibehalten`, wenn die Aktion zum Beenden des Lebenszyklus erfolgreich abgeschlossen `ABANDON` wurde.
5. Beibehaltene Instances warten auf manuelle Maßnahmen. Für Instances in den beibehaltenen Staaten fallen weiterhin die Standardgebühren von Amazon EC2 an. Diese Instances werden nicht auf die gewünschte Kapazität Ihrer Auto Scaling-Gruppe angerechnet, sodass Auto Scaling Ersatz-Instances startet, um die gewünschte Größe beizubehalten. Auto Scaling Scaling-Funktionen wie Instanzaktualisierung und maximale Instanzlebensdauer ignorieren

ebenfalls beibehaltene Instances. Auf diese Weise können Sie Bereinigungsverfahren manuell abschließen, Daten wiederherstellen oder untersuchen, warum das automatische Herunterfahren fehlgeschlagen ist, bevor Sie die Instance manuell beenden.

6. Es erfolgt eine manuelle Kündigung. Nachdem Sie die erforderlichen Aktionen für die beibehaltene Instance abgeschlossen haben, müssen Sie die `TerminateInstanceInAutoScalingGroup` API aufrufen, um die Instance zu beenden.

Konfigurieren Sie die Aufbewahrung von Instanzen

Richten Sie Ihre Amazon EC2 Auto Scaling Scaling-Gruppe so ein, dass Instances beibehalten werden, wenn die Aktionen im Terminierungszyklus fehlschlagen.

Um Instance-Lifecycle-Richtlinien in Ihrer Auto Scaling Scaling-Gruppe zu verwenden, müssen Sie auch einen Termination-Lifecycle-Hook konfigurieren. Wenn Sie eine Instance-Lebenszyklus-Richtlinie konfigurieren, aber keine Terminierungs-Lifecycle-Hooks haben, hat die Richtlinie keine Wirkung. Richtlinien für den Instanzlebenszyklus gelten nur, wenn Aktionen zum Kündigungszyklus abgebrochen werden, nicht, wenn sie erfolgreich mit dem CONTINUE Ergebnis abgeschlossen wurden.

Richtlinien für den Instanzlebenszyklus verwenden Aufbewahrungsauslöser, um zu bestimmen, wann eine Instance aufbewahrt werden soll. Der `TerminateHookAbandon` Auslöser führt in mehreren Szenarien zur Aufbewahrung:

- Wenn Sie die [CompleteLifecycleAction](#) API explizit mit dem ABANDON Ergebnis aufrufen.
- Wenn bei einer Aktion im Terminierungszyklus mit Standardergebnis das ABANDON Timeout überschritten wird, weil das Heartbeat-Timeout erreicht wird, ohne dass ein Heartbeat empfangen wurde.
- Wenn das globale Timeout für eine Aktion im Lebenszyklus einer Kündigung mit einem Standardergebnis erreicht wird ABANDON, das 48 Stunden oder das 100-fache des Heartbeat-Timeouts beträgt, je nachdem, welcher Wert kleiner ist

Console

Um die Aufbewahrung von Instanzen zu konfigurieren

1. Öffnen Sie die Amazon EC2 Auto Scaling Scaling-Konsole

2. Erstellen Sie Ihre Auto Scaling Scaling-Gruppe (die Instance-Lebenszyklus-Richtlinie ist standardmäßig auf Terminate eingestellt)
3. Gehen Sie zu Ihrer Auto Scaling Scaling-Gruppenseite und wählen Sie den Tab Instance Management.
4. Wählen Sie unter Instance-Lebenszyklus-Richtlinie für Lifecycle-Hooks die Option Retain
5. Erstellen Sie Ihre Termination-Lifecycle-Hooks mit:
 - Der Lebenszyklusübergang ist auf „Instanz beenden“ gesetzt
 - Das Standardergebnis ist auf Abandon eingestellt

AWS CLI

Um die Aufbewahrung von Instanzen zu konfigurieren

Verwenden Sie den [create-auto-scaling-group](#) Befehl mit einer Instanzlebenszyklus-Richtlinie:

```
aws autoscaling create-auto-scaling-group \  
--auto-scaling-group-name my-asg \  
--launch-template LaunchTemplateName=my-template,Version='$Latest' \  
--min-size 1 \  
--max-size 3 \  
--desired-capacity 2 \  
--vpc-zone-identifier subnet-12345678 \  
--instance-lifecycle-policy file:///lifecycle-policy.json
```

Inhalt von lifecycle ycle-policy.json:

```
{  
  "RetentionTriggers": {  
    "TerminateHookAbandon": "retain"  
  }  
}
```

So fügen Sie einen Termination-Lebenszyklus-Hook hinzu

Verwenden Sie den [-Befehl:put-lifecycle-hook](#)

```
aws autoscaling put-lifecycle-hook \  

```

```
--lifecycle-hook-name my-termination-hook \  
--auto-scaling-group-name my-asg \  
--lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING \  
--default-result ABANDON \  
--heartbeat-timeout 300
```

Beibehaltene Instanzen verwalten

Überwachen und kontrollieren Sie Amazon EC2 EC2-Instances, die in einen beibehaltenen Zustand versetzt wurden. Verwenden Sie CloudWatch Metriken, um zurückbehaltene Instances nachzuverfolgen, und beenden Sie die zurückgehaltenen Instances dann manuell, nachdem Sie Ihre benutzerdefinierten Aktionen abgeschlossen haben.

Beibehaltene Instances werden nicht auf die gewünschte Kapazität Ihrer Amazon EC2 Auto Scaling Scaling-Gruppe angerechnet. Wenn eine Instance in den Status „Beibehalten“ übergeht, startet Auto Scaling eine Ersatz-Instance, um die gewünschte Kapazität aufrechtzuerhalten. Nehmen wir zum Beispiel an, Ihre Auto Scaling Scaling-Gruppe hat eine gewünschte Kapazität von 10. Wenn eine Instance in den `Terminating:Retained` Status wechselt, startet Auto Scaling eine Ersatz-Instance, um die gewünschte Kapazität von 10 aufrechtzuerhalten. Sie haben jetzt insgesamt 11 laufende Instances: 10 in Ihrer aktiven Gruppe plus 1 beibehaltene Instance. Die Amazon EC2 EC2-Standardgebühren für alle 11 Instances fallen an, bis Sie die beibehaltene Instance manuell kündigen.

Status des Instance-Lebenszyklus der beibehaltenen Instances

Verschaffen Sie sich einen Überblick darüber, wie Instanzen den jeweiligen Lebenszyklusstatus durchlaufen, wenn Richtlinien für den Instanzlebenszyklus verwendet werden. Instances folgen einem bestimmten Pfad von der normalen Kündigung über die Aufbewahrung bis hin zur endgültigen Kündigung.

Wenn die Aufbewahrung ausgelöst wird, durchlaufen Instances den folgenden Status:

1. `Terminating`- Die normale Kündigung beginnt
2. `Terminating:Wait`- Der Lifecycle-Hook wird ausgeführt
3. `Terminating:Proceed`- Lifecycle-Aktionen werden abgeschlossen (unabhängig davon, ob sie erfolgreich waren oder fehlgeschlagen sind)
4. `Terminating:Retained`- Der Hook schlägt fehl, die Instanz wird für manuelles Eingreifen beibehalten

Warm-Pool-Instances nehmen je nach Szenario unterschiedliche Lebenszyklusstatuspfade ein:

Instanzen, die zurück in den Warmpool skalieren:

1. `Warmed:Pending`- Der normale Übergang zum warmen Pool beginnt
2. `Warmed:Pending:Wait`- Der Lifecycle-Hook wird ausgeführt
3. `Warmed:Pending:Proceed`- Lifecycle-Aktionen werden abgeschlossen (unabhängig davon, ob sie erfolgreich waren oder fehlgeschlagen sind)
4. `Warmed:Pending:Retained`- Der Hook schlägt fehl, die Instanz wird für manuelles Eingreifen beibehalten

Instanzen, die vom warmen Pool aus beendet werden:

1. `Warmed:Terminating`- Die normale Kündigung beginnt
2. `Warmed:Terminating:Wait`- Der Lifecycle-Hook wird ausgeführt
3. `Warmed:Terminating:Proceed`- Lifecycle-Aktionen werden abgeschlossen (unabhängig davon, ob sie erfolgreich waren oder fehlgeschlagen sind)
4. `Warmed:Terminating:Retained`- Der Hook schlägt fehl, die Instanz wird für manuelles Eingreifen beibehalten

Überwachen Sie die gespeicherten Instanzen

Da für die Aufbewahrung von Amazon EC2 EC2-Instances Kosten anfallen und manuelles Eingreifen erforderlich sind, ist deren Überwachung unerlässlich. Amazon EC2 Auto Scaling bietet mehrere CloudWatch Metriken zur Nachverfolgung von gespeicherten Instances.

Aktivieren Sie Gruppenmetriken, um beibehaltene Instances nachzuverfolgen:

```
aws autoscaling enable-metrics-collection \  
--auto-scaling-group-name my-asg \  
--metrics GroupTerminatingRetainedInstances
```

Die verfügbaren Metriken sind:

- `GroupTerminatingRetainedInstances` zeigt die Anzahl der Instanzen im `Terminating:Retained` Bundesstaat an.

- `GroupTerminatingRetainedCapacity` zeigt die Kapazitätseinheiten an, die durch Instances im `Terminating:Retained` Bundesstaat repräsentiert werden.
- `WarmPoolTerminatingRetainedCapacity` verfolgt beibehaltene Instances, die im warmen Pool enden.
- `WarmPoolPendingRetainedCapacity` verfolgt die zurückgehaltenen Instanzen, die in den warmen Pool zurückkehren.

Sie können auch die Skalierungsaktivitäten Ihrer Amazon EC2 Auto Scaling-Gruppe überprüfen, um zu verstehen, warum Instances beibehalten wurden. Suchen Sie nach Kündigungsaktivitäten mit `StatusCode: Cancelled` Meldungen zur Begründung des Status, die auf Fehler beim Lifecycle-Hook hinweisen:

```
aws autoscaling describe-scaling-activities \  
--auto-scaling-group-name my-asg
```

Wir empfehlen, CloudWatch Alarmler für diese Metriken zu erstellen, um Sie zu benachrichtigen, wenn Instances in den Status „Beibehalten“ wechseln. Auf diese Weise können Sie die Auswirkungen auf die Kosten verfolgen und sicherstellen, dass Sie nicht vergessen, Instanzen zu bereinigen, die manuelles Eingreifen erfordern.

Beibehaltene Instanzen beenden

Nachdem Sie Ihre benutzerdefinierten Aktionen abgeschlossen haben, beenden Sie Ihre gespeicherten Instances, indem Sie die [TerminateInstanceInAutoScalingGroup](#) API aufrufen:

```
aws autoscaling terminate-instance-in-auto-scaling-group \  
--instance-id i-1234567890abcdef0 \  
--no-should-decrement-desired-capacity
```

Abrufen des Ziellebenszyklus-Status durch Instance-Metadaten

Jede Auto-Scaling-Instance, die Sie starten, durchläuft mehrere Lebenszyklus-Status. Um aus einer Instance heraus benutzerdefinierte Aktionen aufzurufen, die auf bestimmte Lebenszyklusstatus-Übergänge wirken, müssen Sie den Ziellebenszyklus-Status über Instance-Metadaten abrufen.

Beispielsweise benötigen Sie möglicherweise einen Mechanismus, um die Instance-Beendigung innerhalb der Instance zu erkennen, um Code auf der Instance auszuführen, bevor sie beendet

wird. Sie können dies tun, indem Sie Code schreiben, der den Lebenszyklus-Status einer Instance direkt von der Instance aus abfragt. Anschließend können Sie der Auto-Scaling-Gruppe einen Lebenszyklus-Hook hinzufügen, um die Instance so lange am Laufen zu halten, bis Ihr Code den Befehl `complete-lifecycle-action` zum Fortfahren sendet.

Der Lebenszyklus der Auto-Scaling-Instance hat zwei primäre Beharrungszustände – `InService` und `Terminated` – und zwei sekundäre Beharrungszustände – `Detached` und `Standby`. Wenn Sie einen Warm-Pool verwenden, hat der Lebenszyklus vier zusätzliche Beharrungszustände – `Warmed:Hibernated`, `Warmed:Running`, `Warmed:Stopped` und `Warmed:Terminated`.

Wenn sich eine Instance auf den Übergang in einen der vorhergehenden Beharrungszustände vorbereitet, aktualisiert Amazon EC2 Auto Scaling den Wert des Instance-Metadatenelements `autoscaling/target-lifecycle-state`. Um den Ziellebenszyklusstatus innerhalb der Instance abzurufen, müssen Sie den Instance-Metadatendienst verwenden, um ihn aus den Instance-Metadaten abzurufen.

Note

Instance-Metadaten sind Daten über eine Amazon-EC2-Instance, mit denen Anwendungen Instance-Informationen abfragen können. Der Instance-Metadatenservice (IMDS) ist eine On-Instance-Komponente, die von lokalem Code verwendet wird, um auf Instance-Metadaten zuzugreifen. Lokaler Code kann Benutzerdatenskripte oder Anwendungen enthalten, die auf der Instance ausgeführt werden.

Lokaler Code kann mit einer von zwei Methoden auf Instanz-Metadaten von einer laufenden Instanz aus zugreifen: Instanz-Metadatendienst Version 1 (IMDSv1) oder Instanz-Metadatendienst Version 2 (IMDSv2). IMDSv2 verwendet sitzungsorientierte Anfragen und behebt verschiedene Arten von Sicherheitslücken, die für den Zugriff auf die Instanz-Metadaten genutzt werden könnten. Einzelheiten zu diesen beiden Methoden finden Sie unter [Verwendung IMDSv2](#) im Amazon EC2 EC2-Benutzerhandbuch.

IMDSv2

```
[ec2-user ~]$ TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600" ` \  
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-  
data/autoscaling/target-lifecycle-state
```

IMDSv1

```
[ec2-user ~]$ curl http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

Es folgt eine Beispielausgabe.

```
InService
```

Der Ziellebenszyklusstatus ist der Status, in den die Instance wechselt. Der aktuelle Lebenszyklusstatus ist der Status, in dem sich die Instance befindet. Diese können gleich sein, nachdem die Lebenszyklusaktion abgeschlossen ist und die Instance ihren Übergang in den Ziellebenszyklusstatus beendet hat. Sie können den aktuellen Lebenszyklusstatus der Instance nicht aus den Instance-Metadaten abrufen.

Amazon EC2 Auto Scaling begann am 10. März 2022 mit der Generierung des Ziellebenszyklusstatus. Wenn Ihre Instance nach diesem Datum in einen der Ziellebenszyklusstatus wechselt, ist das Ziellebenszyklusstatus-Element in den Instance-Metadaten vorhanden. Andernfalls ist es nicht vorhanden und Sie erhalten einen HTTP-404-Fehler.

Weitere Informationen zum Abrufen von Instance-Metadaten finden Sie unter [Instance-Metadaten abrufen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Ein Tutorial, das Ihnen zeigt, wie Sie einen Lebenszyklus-Hook mit einer benutzerdefinierten Aktion in einem Benutzerdatenskript erstellen, das den Ziel-Lebenszyklusstatus verwendet, finden Sie unter [Tutorial: Verwenden Sie Datenscript- und Instance-Metadaten, um den Lebenszyklusstatus abzurufen](#).

Important

Um sicherzustellen, dass Sie so schnell wie möglich eine benutzerdefinierte Aktion aufrufen können, sollte Ihr lokaler Code IMDS häufig abfragen und es bei Fehlern erneut versuchen.

Fügen Sie Lifecycle-Hooks zu Ihrer Auto Scaling Scaling-Gruppe hinzu

Um Ihre Instances mit automatischer Skalierung in einen Wartezustand zu versetzen und benutzerdefinierte Aktionen für sie durchzuführen, können Sie Ihrer Auto-Scaling-Gruppe Lebenszyklus-Hooks hinzufügen. Benutzerdefinierte Aktionen werden beim Start der Instances oder

vor dem Beenden ausgeführt. Die Instances bleiben in einem Wartezustand, bis Sie entweder die Lebenszyklusaktion beenden oder der Timeout-Zeitraum endet.

Nachdem Sie aus der eine Auto Scaling Scaling-Gruppe erstellt haben AWS-Managementkonsole, können Sie ihr einen oder mehrere Lifecycle-Hooks hinzufügen, bis zu insgesamt 50 Lifecycle-Hooks. Sie können auch das AWS CLI, oder ein SDK verwenden CloudFormation, um einer Auto Scaling Scaling-Gruppe Lifecycle-Hooks hinzuzufügen, während Sie sie erstellen.

Wenn Sie einen Lifecycle-Hook in der Konsole hinzufügen, sendet Amazon EC2 Auto Scaling standardmäßig Lebenszyklusereignisbenachrichtigungen an Amazon EventBridge. Die Verwendung EventBridge eines Benutzerdatenskripts ist eine empfohlene bewährte Methode. Um einen Lifecycle-Hook zu erstellen, der Benachrichtigungen direkt an Amazon SNS oder Amazon SQS sendet, können AWS Lambda Sie den [put-lifecycle-hook](#)Befehl verwenden, wie in den Beispielen in diesem Thema gezeigt.

Inhalt

- [Lebenszyklus-Hooks hinzufügen \(Konsole\)](#)
- [Hinzufügen von Lebenszyklus-Hooks \(AWS CLI\)](#)

Lebenszyklus-Hooks hinzufügen (Konsole)

Gehen Sie folgendermaßen vor, um Ihrer Auto-Scaling-Gruppe einen Lebenszyklus-Hook hinzuzufügen. Um Lebenszyklus-Hooks zum Aufskalieren (Starten von Instances) und Abskalieren (Beenden von Instances oder bei der Rückkehr zu einem warmen Pool) zu erstellen, müssen Sie zwei separate Hooks erstellen.

Bevor Sie beginnen, vergewissern Sie sich, dass Sie bei Bedarf eine benutzerdefinierte Aktion eingerichtet haben, wie unter [Vorbereiten des Hinzufügens eines Lebenszyklus-Hook zu einer Auto-Scaling-Gruppe](#) beschrieben.

So fügen Sie einen Lebenszyklus-Hook für das Aufskalieren hinzu

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe. Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.
3. Wählen Sie auf der Registerkarte Instance management (Instance-Verwaltung) unter Lebenszyklus-Hooks die Option Create Lebenszyklus hook (Lebenszyklus-Hook erstellen) aus.

4. Gehen Sie wie folgt vor, um einen Lebenszyklus-Hook zum Aufskalieren (Start von Instances) zu definieren:
 - a. Geben Sie bei Lebenszyklus hook name (Name des Lebenszyklus-Hooks) einen Namen für den Lebenszyklus-Hook an.
 - b. Wählen Sie bei Lifecycle Transition (Lebenszykluswechsel) die Option Instance launch (Instance-Start) aus.
 - c. Geben Sie für Heartbeat-Zeitüberschreitung die Zeitspanne in Sekunden an, für die Instances in einem Wartezustand verbleiben sollen, wenn Sie aufskalieren, bevor die Zeitüberschreitung des Hook erreicht ist. Der Bereich liegt zwischen 30 und 7200 Sekunden. Das Festlegen eines langen Timeout-Zeitraums bietet mehr Zeit für den Abschluss der benutzerdefinierten Aktion. Wenn Sie dann vor Ablauf des Timeout-Zeitraums fertig sind, senden Sie den [complete-lifecycle-action](#) Befehl, damit die Instance zum nächsten Status übergehen kann.
 - d. Definieren Sie für ein Default result (Standardergebnis) die Aktion, die ausgeführt werden soll, wenn für den Lebenszyklus-Hook eine Zeitüberschreitung oder ein unerwarteter Fehler auftritt. Sie können entweder FORTSETZEN oder ABBRUCH auswählen.
 - Wenn Sie FORTSETZEN wählen, kann die Auto-Scaling-Gruppe mit allen anderen Lebenszyklus-Hooks fortfahren und die Instance dann in Betrieb nehmen.
 - Wenn Sie ABBRUCH wählen, beendet die Auto-Scaling-Gruppe alle verbleibenden Aktionen und beendet die Instance sofort.
 - e. (Optional) Geben Sie bei Benachrichtigungsmetadaten die weiteren Informationen an, die Sie hinzufügen möchten, wenn Amazon EC2 Auto Scaling eine Nachricht an das Benachrichtigungsziel sendet.
5. Wählen Sie Erstellen aus.

So fügen Sie einen Lebenszyklus-Hook für das Abskalieren hinzu

1. Wählen Sie Lebenszyklus-Hook erstellen, um dort weiterzumachen, wo Sie aufgehört haben, nachdem Sie einen Lebenszyklus-Hook für die horizontale Aufskalierung erstellt haben.
2. Gehen Sie wie folgt vor, um einen Lebenszyklus-Hook für die Abskalierung zu definieren (Instances, die beendet werden oder zu einem warmen Pool zurückkehren):
 - a. Geben Sie bei Lebenszyklus hook name (Name des Lebenszyklus-Hooks) einen Namen für den Lebenszyklus-Hook an.

- b. Wählen Sie bei Lifecycle Transition (Lebenszykluswechsel) die Option Instance Terminate (Instance-Beendigung) aus.
 - c. Geben Sie für Heartbeat-Zeitüberschreitung die Zeitspanne in Sekunden an, für die Instances in einem Wartezustand verbleiben sollen, wenn Sie aufskalieren, bevor die Zeitüberschreitung des Hook erreicht ist. Wir empfehlen ein kurzes Timeout von 30 zwei 120 Sekunden, je nachdem, wie viel Zeit Sie für die Ausführung der letzten Aufgaben benötigen, wie z. B. das Abrufen von EC2-Protokollen. CloudWatch
 - d. Geben Sie als Default Result (Standardergebnis) die Aktion an, die von der Auto-Scaling-Gruppe ausgeführt wird, wenn für den Lebenszyklus-Hook ein Timeout oder ein unerwarteter Fehler auftritt. Sowohl ABANDON (ABBRUCH) als auch CONTINUE (FORTSETZEN) ermöglichen das Beenden der Instance.
 - Wenn Sie CONTINUE (FORTSETZEN) wählen, kann die Auto-Scaling-Gruppe vor der Beendigung alle verbleibenden Aktionen, z. B. andere Lebenszyklus-Hooks, ausführen.
 - Wenn Sie ABRUCH wählen, beendet die Auto-Scaling-Gruppe die Instance sofort.
 - e. (Optional) Geben Sie bei Benachrichtigungsmetadaten die weiteren Informationen an, die Sie hinzufügen möchten, wenn Amazon EC2 Auto Scaling eine Nachricht an das Benachrichtigungsziel sendet.
3. Wählen Sie Erstellen aus.

Hinzufügen von Lebenszyklus-Hooks (AWS CLI)

Erstellen und aktualisieren Sie Lebenszyklus-Hooks über den Befehl [put-lifecycle-hook](#) .

Verwenden Sie den folgenden Befehl zum Ausführen einer Aktion bei einer horizontalen Skalierung nach oben:

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

Verwenden Sie hingegen den folgenden Befehl, um bei einer horizontalen Skalierung nach unten eine Aktion durchzuführen:

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING
```

Um Benachrichtigungen mit Amazon SNS oder Amazon SQS zu empfangen, fügen Sie die Optionen `--notification-target-arn` und `--role-arn` hinzu. Um Benachrichtigungen über zu erhalten AWS Lambda, fügen Sie den hinzu. `--notification-target-arn`

Im folgenden Beispiel wird ein Lebenszyklus-Hook erstellt, der ein SNS-Thema mit dem Namen *my-sns-topic* als Benachrichtigungsziel definiert.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING \  
  --notification-target-arn arn:aws:sns:region:123456789012:my-sns-topic \  
  --role-arn arn:aws:iam::123456789012:role/my-notification-role
```

Das Thema erhält eine Testbenachrichtigung mit dem folgenden Schlüssel-Wert-Paar:

```
"Event": "autoscaling:TEST_NOTIFICATION"
```

Standardmäßig erstellt der [put-lifecycle-hook](#) Befehl einen Lifecycle-Hook mit einem Heartbeat-Timeout von 3600 Sekunden (eine Stunde).

Um das Heartbeat-Timeout für einen vorhandenen Lebenszyklus-Hook zu ändern, fügen Sie die Option `--heartbeat-timeout` hinzu, wie im folgenden Beispiel gezeigt.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \  
  --auto-scaling-group-name my-asg --heartbeat-timeout 120
```

Wenn sich eine Instance bereits im Wartestatus befindet, können Sie verhindern, dass der Lifecycle-Hook das Timeout überschreitet, indem Sie mit dem [record-lifecycle-action-heartbeat](#) CLI-Befehl einen Heartbeat aufzeichnen. Diese erweitert die Zeitüberschreitung um den Zeitüberschreitungswert, den Sie bei der Erstellung des Lebenszyklus-Hooks festgelegt haben. Wenn Sie vor Ablauf des Timeout-Zeitraums fertig sind, können Sie den [complete-lifecycle-action](#) CLI-Befehl senden, damit die Instance zum nächsten Status übergehen kann. Weitere Informationen und Beispiele finden Sie unter [Eine Lebenszyklusaktion in einer Auto Scaling Scaling-Gruppe abschließen](#).

Eine Lebenszyklusaktion in einer Auto Scaling Scaling-Gruppe abschließen

Reagiert eine Auto-Scaling-Gruppe auf ein Lebenszyklus-Ereignis, versetzt sie die Instance in einen Wartestatus und sendet eine Ereignisbenachrichtigung. Sie können eine benutzerdefinierte Aktion ausführen, während sich die Instance in einem Wartestatus befindet.

Das Abschließen der Lebenszyklus-Aktion mit dem Ergebnis von CONTINUE ist hilfreich, wenn Sie den Vorgang vor Ablauf des Timeouts beenden. Wenn Sie die Lebenszyklus-Aktion nicht abschließen, nimmt der Lebenszyklus-Hook nach Ablauf des Timeout-Zeitraums den Status an, den Sie als Standardergebnis angegeben haben.

Inhalt

- [Eine Lebenszyklus-Aktion abschließen \(manuell\)](#)
- [Eine Lebenszyklus-Aktion abschließen \(automatisch\)](#)

Eine Lebenszyklus-Aktion abschließen (manuell)

Das folgende Verfahren gilt für die Befehlszeilenschnittstelle und wird in der Konsole nicht unterstützt. Die zu ersetzenden Informationen wie die Instance-ID oder der Name einer Auto-Scaling-Gruppe werden kursiv dargestellt.

So führen Sie eine Lebenszyklus-Aktion aus (AWS CLI)

1. Falls Sie mehr Zeit für die benutzerdefinierte Aktion benötigen, verwenden Sie den Befehl [record-lifecycle-action-heartbeat](#), um die Zeit für die Zeitüberschreitung zurückzusetzen und die Instance im Wartestatus zu belassen. Beträgt der Zeitüberschreitungszeitraum z. B. eine Stunde und Sie rufen diesen Befehl nach 30 Minuten auf, verbleibt die Instance für eine zusätzliche Stunde bzw. insgesamt 90 Minuten in einem Wartestatus.

Sie können das Token der Lebenszyklusaktion das Sie mit der [Benachrichtigung](#) erhalten haben, wie im folgenden Befehl gezeigt angeben.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Alternativ können Sie auch die ID der Instance angeben, die Sie mit der [Benachrichtigung](#) erhalten haben, wie im folgenden Befehl gezeigt.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg --instance-id i-1a2b3c4d
```

2. Wenn Sie die benutzerdefinierte Aktion vor Ablauf des Timeout-Zeitraums beenden, verwenden Sie den [complete-lifecycle-action](#)Befehl, damit die Auto Scaling Scaling-Gruppe die Instance weiter starten oder beenden kann. Sie können das Token für die Lebenszyklusaktion wie im folgenden Befehl angeben:

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \  
--lifecycle-hook-name my-launch-hook --auto-scaling-group-name my-asg \  
--lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Alternativ können Sie die ID der Instance wie im folgenden Befehl angeben:

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \  
--instance-id i-1a2b3c4d --lifecycle-hook-name my-launch-hook \  
--auto-scaling-group-name my-asg
```

Eine Lebenszyklus-Aktion abschließen (automatisch)

Wenn Sie ein Skript mit Benutzerdaten haben, das Ihre Instances nach dem Start konfiguriert, müssen Sie die Lebenszyklusaktionen nicht manuell durchführen. Sie können den [complete-lifecycle-action](#)Befehl dem Skript hinzufügen. Das Skript kann die Instance-ID aus den Instance-Metadaten abrufen und Amazon EC2 Auto Scaling signalisieren, wenn die Bootstrap-Skripte erfolgreich abgeschlossen wurden.

Wenn Sie nicht bereits dabei sind, aktualisieren Sie das Skript, sodass es die Instance-ID der Instance aus den Instance-Metadaten abrufen. Weitere Informationen finden Sie unter [Instance-Metadaten abrufen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Wenn Sie Lambda verwenden, können Sie auch einen Rückruf im Code Ihrer Funktion einrichten, damit der Lebenszyklus der Instance fortgesetzt werden kann, wenn die benutzerdefinierte Aktion erfolgreich ist. Weitere Informationen finden Sie unter [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#).

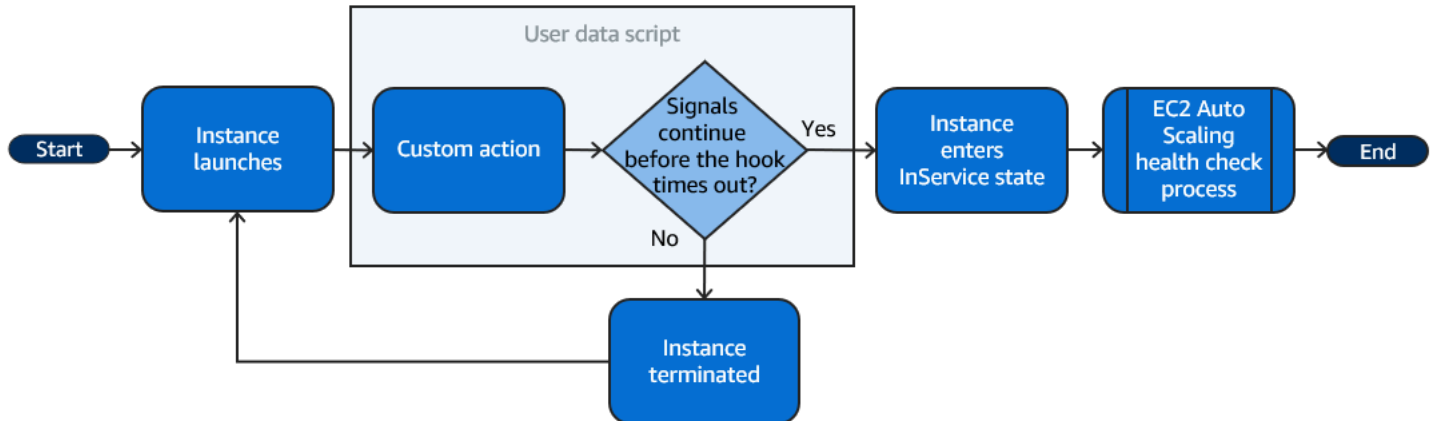
Tutorial: Verwenden Sie Datenskript- und Instance-Metadaten, um den Lebenszyklusstatus abzurufen

Eine gängige Methode, benutzerdefinierte Aktionen für Lifecycle-Hooks zu erstellen, ist die Verwendung von Benachrichtigungen, die Amazon EC2 Auto Scaling an andere Dienste wie Amazon EventBridge sendet. Sie können jedoch vermeiden, dass Sie zusätzliche Infrastruktur erstellen

müssen, indem Sie stattdessen ein Benutzerdatenskript verwenden, um den Code, der Instances konfiguriert und die Lebenszyklusaktion abschließt, in die Instances selbst zu verschieben.

Das folgende Tutorial veranschaulicht die Verwendung eines Benutzerdatenskripts und Instance-Metadaten. Sie erstellen eine grundlegende Auto-Scaling-Gruppenkonfiguration mit einem Benutzerdatenskript, das die [Ziel-Lebenszyklusstatus](#) der Instances in Ihrer Gruppe liest und eine Rückrufaktion in einer bestimmten Phase des Lebenszyklus einer Instance ausführt, um den Startprozess fortzusetzen.

Die folgende Abbildung fasst den Ablauf für ein Scale-Out-Ereignis zusammen, wenn Sie ein Benutzerdatenskript verwenden, um eine benutzerdefinierte Aktion auszuführen. Nach dem Start einer Instance wird der Lebenszyklus der Instance angehalten, bis der Lifecycle-Hook abgeschlossen ist, entweder durch eine Zeitüberschreitung oder dadurch, dass Amazon EC2 Auto Scaling ein Signal zum Fortfahren empfängt.



Inhalt

- [Schritt 1: Erstellen einer IAM-Rolle mit Berechtigungen zum Abschließen von Lebenszyklus-Aktionen](#)
- [Schritt 2: Erstellen Sie eine Startvorlage und schließen Sie die IAM-Rolle und ein Benutzerdatenskript ein](#)
- [Schritt 3: Erstellen einer Auto-Scaling-Gruppe](#)
- [Schritt 4: Hinzufügen eines Lebenszyklus-Hooks](#)
- [Schritt 5: Testen und Prüfen der Funktionalität](#)
- [Schritt 6: Bereinigen](#)
- [Zugehörige Ressourcen](#)

Schritt 1: Erstellen einer IAM-Rolle mit Berechtigungen zum Abschließen von Lebenszyklus-Aktionen

Wenn Sie das AWS CLI oder ein AWS SDK verwenden, um einen Rückruf zu senden, um Lebenszyklusaktionen abzuschließen, müssen Sie eine IAM-Rolle mit Berechtigungen zum Abschließen von Lebenszyklusaktionen verwenden.

So erstellen Sie die Richtlinie

1. Öffnen Sie in der IAM-Konsole [Policies \(Richtlinien\)](#) und wählen Sie dann Create policy (Richtlinie erstellen) aus.
2. Wählen Sie den Tab JSON.
3. Kopieren Sie das folgende Richtliniendokument und fügen Sie es in das Feld Policy Document (Richtliniendokument) ein. Ersetzen Sie das *sample text* durch Ihre Kontonummer und den Namen der Auto Scaling Scaling-Gruppe, die Sie erstellen möchten (**TestAutoScalingEvent-group**).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
        "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/TestAutoScalingEvent-group"
    }
  ]
}
```

4. Wählen Sie Weiter aus.
5. Geben Sie unter Policy name (Richtliniennamen) **TestAutoScalingEvent-policy** ein. Wählen Sie Richtlinie erstellen aus.

Wenn Sie die Richtlinie fertig erstellt haben, können Sie eine Rolle erstellen, die sie verwendet.

So erstellen Sie die Rolle

1. Wählen Sie im Navigationsbereich auf der linken Seite Roles (Rollen).
2. Wählen Sie Rolle erstellen aus.
3. Wählen Sie für Select trusted entity (Vertrauenswürdige Entität auswählen) die Option AWS - Service.
4. Wählen Sie für Ihren Anwendungsfall die Option EC2 und dann Next (Weiter) aus.
5. Wählen Sie unter Berechtigungen hinzufügen die Richtlinie aus, die Sie erstellt haben (TestAutoScalingEvent-policy). Wählen Sie anschließend Weiter.
6. Geben Sie auf der Seite Set role name and review (Rollenname festlegen und überprüfen) für Role name (Rollenname) **TestAutoScalingEvent-role** ein und wählen Sie Create role (Rolle erstellen) aus.

Schritt 2: Erstellen Sie eine Startvorlage und schließen Sie die IAM-Rolle und ein Benutzerdatenskript ein

Erstellen Sie eine Startvorlage für die Verwendung mit einer Auto-Scaling-Gruppe. Fügen Sie die von Ihnen erstellte IAM-Rolle und das bereitgestellte Beispielbenutzerdatenskript ein.

Eine Startvorlage erstellen

1. Öffnen Sie die Seite [Startvorlagen](#) der Amazon EC2 Konsole.
2. Wählen Sie Startvorlage erstellen.
3. Geben Sie für Startvorlagenname **TestAutoScalingEvent-template** ein.
4. Unter Auto-Scaling-Anleitung aktivieren Sie das Kontrollkästchen.
5. Wählen Sie für Application and OS Images (Amazon Machine Image) (Anwendungs- und Betriebssystem-Images (Amazon Machine Image)) Amazon Linux 2 (HVM), SSD-Volume-Typ, 64-Bit (x86) aus der Quick Start(Schnellstart)-Liste.
6. Wählen Sie für Instance type (Instance-Typ) einen Typ von Amazon-EC2-Instance (z. B. „t2.micro“) aus.
7. Für Erweiterte Details erweitern Sie den Abschnitt, um die Felder anzuzeigen.
8. Wählen Sie für das IAM-Instanzprofil den IAM-Instanzprofilnamen Ihrer IAM-Rolle (-role) aus. TestAutoScalingEvent Ein Instance-Profil ist ein Container für eine IAM-Rolle, mit dem Amazon EC2 einer Instance die IAM-Rolle übergibt, wenn die Instance gestartet wird.

Wenn Sie eine IAM-Rolle mithilfe der IAM-Konsole erstellt haben, hat die Konsole automatisch ein Instance-Profil mit demselben Namen wie der entsprechenden Rolle erzeugt.

9. Kopieren Sie für User date (Benutzerdaten) das folgende Beispiel-Benutzerdatenskript und fügen Sie es in das Feld ein. Ersetzen Sie den Beispieltext für `group_name` durch den Namen der Auto Scaling Scaling-Gruppe, die Sie erstellen möchten, und `region` durch den Namen, den AWS-Region Ihre Auto Scaling Scaling-Gruppe verwenden soll.

```
#!/bin/bash

function token {
    echo "X-aws-ec2-metadata-token: $(curl -X PUT 'http://169.254.169.254/latest/api/token' -H 'X-aws-ec2-metadata-token-ttl-seconds: 21600')"
}

function get_target_state {
    echo $(curl -H "$(token)" -s http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state)
}

function get_instance_id {
    echo $(curl -H "$(token)" -s http://169.254.169.254/latest/meta-data/instance-id)
}

function complete_lifecycle_action {
    instance_id=$(get_instance_id)
    group_name='TestAutoScalingEvent-group'
    region='us-west-2'

    echo $instance_id
    echo $region
    echo $(aws autoscaling complete-lifecycle-action \
        --lifecycle-hook-name TestAutoScalingEvent-hook \
        --auto-scaling-group-name $group_name \
        --lifecycle-action-result CONTINUE \
        --instance-id $instance_id \
        --region $region)
}

function main {
    while true
```

```
do
    target_state=$(get_target_state)
    if [ \"$target_state\" = \"InService\" ]; then
        # Change hostname
        export new_hostname=\"${group_name}-${instance_id}\"
        hostname $new_hostname
        # Send callback
        complete_lifecycle_action
        break
    fi
    echo $target_state
    sleep 5
done
}

main
```

Dieses einfache Benutzerdatenskript führt folgende Aktionen aus:

- Ruft die Instance-Metadaten auf, um den Ziellebenszyklusstatus und die Instance-ID aus den Instance-Metadaten abzurufen
- Ruft den Ziellebenszyklusstatus wiederholt ab, bis er sich auf InService ändert
- Ändert den Hostnamen der Instance in die Instance-ID, der dem Namen der Auto-Scaling-Gruppe vorangestellt ist, wenn der Ziellebenszyklusstatus InService lautet
- Sendet einen Rückruf durch Aufruf des CLI-Befehls complete-lifecycle-action, um Amazon EC2 Auto Scaling zu signalisieren, den EC2-Startprozess zu CONTINUE

10. Wählen Sie Startvorlage erstellen.

11. Wählen Sie auf der Bestätigungsseite Create Auto Scaling group (Auto-Scaling-Gruppe erstellen) aus.

Note

Weitere Beispiele, die Sie als Referenz für die Entwicklung Ihres Benutzerdatenskripts verwenden können, finden Sie im [GitHub Repository](#) für Amazon EC2 Auto Scaling.

Schritt 3: Erstellen einer Auto-Scaling-Gruppe

Nachdem Sie die Startvorlage erstellt haben, erstellen Sie eine Auto-Scaling-Gruppe.

So erstellen Sie eine Auto Scaling-Gruppe

1. Geben Sie auf der Seite Choose launch template or configuration (Startvorlage oder -konfiguration auswählen) für Auto Scaling group name (Auto-Scaling-Gruppenname) einen Namen für Ihre Auto-Scaling-Gruppe ein (**TestAutoScalingEvent-group**).
2. Klicken Sie auf Next (Weiter), um die Seite Choose instance launch options (Wählen Sie Instance-Startoptionen) aufzurufen.
3. Wählen Sie unter Network (Netzwerk) eine VPC aus.
4. Wählen Sie für Availability Zones and subnets (Availability Zones und Subnetze) ein oder mehrere Subnetze aus einer oder mehreren Availability Zones aus.
5. Verwenden Sie im Abschnitt Instance type requirements (Anforderungen an den Instance-Typ) die Standardeinstellung, um diesen Schritt zu vereinfachen. (Setzen Sie die Startvorlage nicht außer Kraft.) In diesem Tutorial werden Sie nur eine On-Demand-Instance mit dem in Ihrer Startvorlage angegebenen Instance-Typ starten.
6. Wählen Sie unten auf dem Bildschirm Überspringen zum Review.
7. Überprüfen Sie auf der Seite Review (Prüfen) die Details Ihrer Auto-Scaling-Gruppe und wählen Sie dann Create Auto Scaling group (Auto-Scaling-Gruppe erstellen).

Schritt 4: Hinzufügen eines Lebenszyklus-Hooks

Fügen Sie einen Lebenszyklus-Hook hinzu, um die Instance in einem Wartezustand zu halten, bis Ihre Lebenszyklusaktion abgeschlossen ist.

So fügen Sie einen Lebenszyklus-Hook hinzu

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe. Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.
3. Wählen Sie im unteren Bereich auf der Registerkarte Instance management (Instance-Verwaltung) unter Lebenszyklus-Hooks die Option Create Lebenszyklus hook (Lebenszyklus-Hook erstellen) aus.

4. Gehen Sie wie folgt vor, um einen Lebenszyklus-Hook zum Aufskalieren (Start von Instances) zu definieren:
 - a. Geben Sie für Lebenszyklus-Hooks den Wert **TestAutoScalingEvent-hook** ein.
 - b. Wählen Sie bei Lifecycle Transition (Lebenszykluswechsel) die Option Instance launch (Instance-Start) aus.
 - c. Geben Sie für Heartbeat timeout (Heartbeat-Zeitüberschreitung) den Wert **300** für die Anzahl der Sekunden ein, die auf einen Rückruf von Ihrem Benutzerdatenskript gewartet werden soll.
 - d. Für Standardergebnis wählen Sie ABBRECHEN aus. Wenn die Zeitüberschreitung des Hooks erreicht ist, ohne einen Rückruf von Ihrem Benutzerdatenskript erhalten zu haben, beendet die Auto-Scaling-Gruppe die neue Instance.
 - e. (Optional) Halten Sie Notification metadata (Benachrichtigungs-Metadaten) frei.
5. Wählen Sie Erstellen aus.

Schritt 5: Testen und Prüfen der Funktionalität

Um die Funktionalität zu testen, aktualisieren Sie die Auto-Scaling-Gruppe, indem Sie die gewünschte Kapazität der Auto-Scaling-Gruppe um 1 erhöhen. Das Benutzerdatenskript wird ausgeführt und überprüft den Ziellebenszyklusstatus der Instance kurz nach dem Start der Instance. Das Skript ändert den Hostnamen und sendet eine Rückrufaktion, wenn der Ziellebenszyklusstatus InService ist. Dieser Vorgang dauert normalerweise nur ein paar Sekunden.

So erhöhen Sie die Größe der Auto-Scaling-Gruppe

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe. Zeigen Sie Details in einem unteren Bereich an, während Sie weiterhin die oberen Zeilen des oberen Bereichs sehen.
3. Wählen Sie im unteren Bereich auf der Registerkarte Details die Option Gruppendetails, Bearbeiten aus.
4. Erhöhen Sie für Desired capacity (Gewünschte Kapazität den aktuellen Wert um 1.
5. Wählen Sie Aktualisieren aus. Während eine Instance gestartet wird, zeigt die Spalte Status den Status Updating capacity (Kapazität aktualisieren) an.

Nachdem Sie die gewünschte Kapazität erhöht haben, können Sie überprüfen, ob die Instance erfolgreich gestartet wurde und nicht von der Beschreibung der Skalierungsaktivitäten beendet wurde.

Ansehen der Skalierungsaktivität

1. Wählen Sie auf der Seite Auto-Scaling-Gruppen Ihre Gruppe aus.
2. Auf der Registerkarte Activity (Aktivität) wird unter Activity history (Aktivitätsverlauf) in der Spalte Status angezeigt, ob Ihre Auto-Scaling-Gruppe Instances erfolgreich gestartet hat.
3. Wenn das Benutzerdatenskript fehlschlägt, sehen Sie nach Ablauf des Timeout-Zeitraums eine Skalierungsaktivität mit dem Status Canceled und eine Statusmeldung `Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result.`

Schritt 6: Bereinigen

Wenn Sie mit den Ressourcen gearbeitet haben, die Sie für dieses Tutorial erstellt haben, führen Sie die folgenden Schritte aus, um sie zu löschen.

So löschen Sie den Lebenszyklus-Hook

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.
3. Wählen Sie auf der Registerkarte Instance management (Instance-Verwaltung) unter Lebenszyklus-Hooks den Lebenszyklus (`TestAutoScalingEvent-hook`) aus.
4. Wählen Sie Actions (Aktionen), Delete (Löschen) aus.
5. Um dies zu bestätigen, wählen Sie erneut Delete (Löschen) aus.

Löschen Sie die Startvorlage wie folgt:

1. Öffnen Sie die Seite [Startvorlagen](#) der Amazon EC2 Konsole.
2. Wählen Sie Ihre Startvorlage (`TestAutoScalingEvent-template`) und anschließend Actions (Aktionen) und Delete template (Vorlage löschen) aus.
3. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **Delete** ein, um das Löschen der angegebenen Auto-Scaling-Gruppe zu bestätigen, wählen Sie dann Löschen.

Wenn Sie mit der Beispiel-Auto-Scaling-Gruppe fertig sind, löschen Sie sie. Sie können auch die von Ihnen erstellte IAM-Rolle und Berechtigungsrichtlinie löschen.

Löschen Sie die Auto-Scaling-Gruppe wie folgt:

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe (TestAutoScalingEvent-group) und wählen Sie Delete (Löschen) aus.
3. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **delete** ein, um das Löschen der angegebenen Auto-Scaling-Gruppe zu löschen, wählen Sie dann Löschen.

Ein Ladesymbol in der Spalte Name zeigt an, dass die Auto-Scaling-Gruppe gelöscht wird. Es dauert einige Minuten, bis die Instances beendet werden und die Gruppe gelöscht wird.

Löschen Sie die IAM-Rolle wie folgt:

1. Öffnen Sie die Seite [Roles \(Rollen\)](#) in der IAM-Konsole.
2. Wählen Sie die Rolle der Funktion (TestAutoScalingEvent-role).
3. Wählen Sie Löschen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie den Namen der Rolle ein und wählen Sie dann Delete (Löschen).

Löschen der IAM-Richtlinie

1. Öffnen Sie die Seite [Richtlinien](#) in der IAM-Konsole.
2. Wählen Sie die Richtlinie aus, die Sie erstellt haben (TestAutoScalingEvent-policy).
3. Wählen Sie Aktionen, Löschen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie den Namen der Richtlinie ein und wählen Sie dann Delete (Löschen).

Zugehörige Ressourcen

Die folgenden damit verbundenen Themen können hilfreich sein, wenn Sie Code entwickeln, der auf der Grundlage der in den Instance-Metadaten verfügbaren Daten Aktionen für Instances aufruft.

- [Abrufen des Ziellebenszyklus-Status durch Instance-Metadaten](#). In diesem Abschnitt wird der Lebenszyklus-Status für andere Anwendungsfälle, wie z. B. die Beendigung der Instance, beschrieben.
- [Lebenszyklus-Hooks hinzufügen \(Konsole\)](#). Diese Prozedur zeigt Ihnen, wie Sie Lebenszyklus-Hooks sowohl für Aufskalieren (Start von Instances) als auch Abskalieren (Beendigung von Instances oder Rückkehr zu einem warmen Pool) hinzufügen können.
- [Kategorien von Instance-Metadaten](#) im Amazon EC2 EC2-Benutzerhandbuch. In diesem Thema sind alle Kategorien von Instance-Metadaten aufgeführt, mit denen Sie Aktionen auf EC2-Instances aufrufen können.

Ein Tutorial, das Ihnen zeigt, wie Sie Amazon verwenden, um Regeln EventBridge zu erstellen, die Lambda-Funktionen auf der Grundlage von Ereignissen aufrufen, die mit den Instances in Ihrer Auto Scaling Scaling-Gruppe passieren, finden Sie unter [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#)

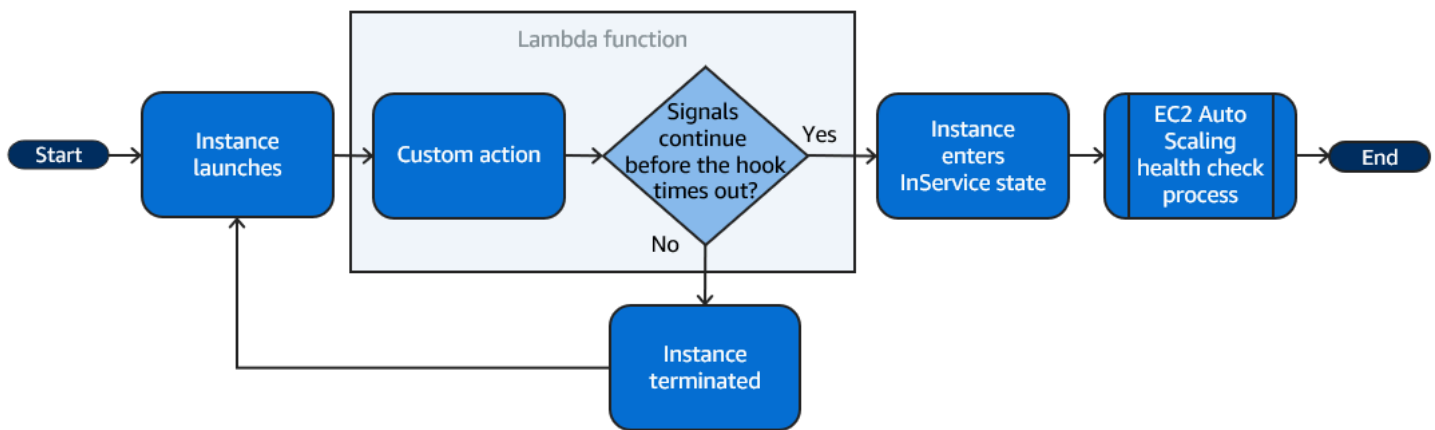
Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft

In dieser Übung erstellen Sie eine EventBridge Amazon-Regel, die ein Filtermuster enthält, das bei Übereinstimmung eine AWS Lambda Funktion als Regelziel aufruft. Wir stellen das Filtermuster und Beispielfunktionscode zur Verfügung.

Wenn alles richtig konfiguriert ist, führt die Lambda-Funktion am Ende dieses Tutorials beim Start von Instances eine benutzerdefinierte Aktion aus. Die benutzerdefinierte Aktion protokolliert einfach das Ereignis im CloudWatch Logs-Protokollstream, der der Lambda-Funktion zugeordnet ist.

Die Lambda-Funktion führt auch einen Rückruf durch, damit der Lebenszyklus der Instance fortgesetzt werden kann, wenn diese Aktion erfolgreich ist. Die Instance kann jedoch den Start abbrechen und beendet werden, wenn die Aktion fehlschlägt.

In der folgenden Abbildung wird der Ablauf für ein Scale-Out-Ereignis zusammengefasst, wenn Sie eine Lambda-Funktion verwenden, um eine benutzerdefinierte Aktion auszuführen. Nach dem Start einer Instance wird der Lebenszyklus der Instance angehalten, bis der Lifecycle-Hook abgeschlossen ist, entweder durch eine Zeitüberschreitung oder dadurch, dass Amazon EC2 Auto Scaling ein Signal zum Fortfahren empfängt.



Note

Je nach Anwendungsfall können Sie einen Lifecycle-Hook konfigurieren, indem Sie die folgenden Schritte ausführen und eine Regel erstellen. EventBridge Oder Sie können eine Lambda-Funktion verwenden, um einen Lifecycle-Hook direkt zu konfigurieren, ohne eine EventBridge Regel zu erstellen.

Inhalt

- [Voraussetzungen](#)
- [Schritt 1: Erstellen einer IAM-Rolle mit Berechtigungen zum Abschließen von Lebenszyklus-Aktionen](#)
- [Schritt 2: Erstellen einer Lambda-Funktion](#)
- [Schritt 3: Erstellen Sie eine Regel EventBridge](#)
- [Schritt 4: Hinzufügen eines Lebenszyklus-Hooks](#)
- [Schritt 5: Testen und Prüfen des Ereignisses](#)
- [Schritt 6: Bereinigen](#)
- [Zugehörige Ressourcen](#)

Voraussetzungen

Erstellen Sie vor Beginn dieses Tutorials eine Auto-Scaling-Gruppe, falls noch keine vorhanden ist. Öffnen Sie zum Erstellen einer Auto-Scaling-Gruppe die Seite [Auto-Scaling-Gruppen](#) der Amazon EC2-Konsole und wählen Sie Eine Auto-Scaling-Gruppe erstellen aus.

Schritt 1: Erstellen einer IAM-Rolle mit Berechtigungen zum Abschließen von Lebenszyklus-Aktionen

Bevor Sie eine Lambda-Funktion erstellen, müssen Sie zunächst eine Ausführungsrolle und eine Berechtigungsrichtlinie erstellen, damit Lambda Lebenszyklus-Hooks abschließen kann.

So erstellen Sie die Richtlinie

1. Öffnen Sie in der IAM-Konsole [Policies \(Richtlinien\)](#) und wählen Sie dann Create policy (Richtlinie erstellen) aus.
2. Wählen Sie den Tab JSON.
3. Fügen Sie im Feld Richtliniendokument das folgende Richtliniendokument in das Feld ein und ersetzen Sie den Text durch Ihre Kontonummer und den Namen Ihrer Auto Scaling Scaling-Gruppe. *italics*

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
        "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/my-asg"
    }
  ]
}
```

4. Wählen Sie Weiter aus.
5. Geben Sie unter Policy name (Richtliniennamen) **LogAutoScalingEvent-policy** ein. Wählen Sie Richtlinie erstellen aus.

Wenn Sie die Richtlinie fertig erstellt haben, können Sie eine Rolle erstellen, die sie verwendet.

So erstellen Sie die Rolle

1. Wählen Sie im Navigationsbereich auf der linken Seite Roles (Rollen).
2. Wählen Sie Rolle erstellen aus.
3. Wählen Sie für Select trusted entity (Vertrauenswürdige Entität auswählen) die Option AWS - Dienst.
4. Wählen Sie für Ihren Anwendungsfall Lambda und dann Next (Weiter) aus.
5. Wählen Sie unter Berechtigungen hinzufügen die Richtlinie aus, die Sie erstellt haben (LogAutoScalingEvent-policy), und die benannte AWSLambdaBasicExecutionRoleRichtlinie aus. Wählen Sie anschließend Weiter.

Note

Die AWSLambdaBasicExecutionRoleRichtlinie verfügt über die Berechtigungen, die die Funktion benötigt, um Protokolle in Logs zu CloudWatch schreiben.

6. Geben Sie auf der Seite Set role name and review (Rollenname festlegen und überprüfen) für Role name (Rollenname) **LogAutoScalingEvent-role** ein und wählen Sie Create role (Rolle erstellen) aus.

Schritt 2: Erstellen einer Lambda-Funktion

Erstellen Sie eine Lambda-Funktion, die als Ziel für Ereignisse dienen soll. Die in Node.js geschriebene Lambda-Beispielfunktion wird aufgerufen, EventBridge wenn ein entsprechendes Ereignis von Amazon EC2 Auto Scaling ausgelöst wird.

So erstellen Sie eine Lambda-Funktion

1. Öffnen Sie die Seite [Funktionen](#) der Lambda-Konsole.
2. Wählen Sie Funktion erstellen und Von Grund auf neu erstellen aus.
3. Geben Sie unter Basic Information (Grundlegende Informationen) für Function name (Funktionsname) **LogAutoScalingEvent** ein.
4. Wählen Sie unter Laufzeit die Option Node.js 18.x aus.
5. Scrollen Sie nach unten und wählen Sie Ändern der standardmäßigen Ausführungsrolle und dann unter Ausführungsrolle Verwenden einer vorhandenen Rolle aus.
6. Wählen Sie für Existing role die Option -role aus. LogAutoScalingEvent

7. Übernehmen Sie im Übrigen die Standardwerte.
8. Wählen Sie Funktion erstellen. Sie kehren zum Code und zur Konfiguration der Funktion zurück.
9. Fügen Sie bei geöffneter LogAutoScalingEvent-Funktion in der Konsole unter Code-Quelle im Editor den folgenden Beispielcode in die Datei index.mjs ein.

```
import { AutoScalingClient, CompleteLifecycleActionCommand } from "@aws-sdk/client-auto-scaling";
export const handler = async(event) => {
  console.log('LogAutoScalingEvent');
  console.log('Received event:', JSON.stringify(event, null, 2));
  var autoscaling = new AutoScalingClient({ region: event.region });
  var eventDetail = event.detail;
  var params = {
    AutoScalingGroupName: eventDetail['AutoScalingGroupName'], /* required */
    LifecycleActionResult: 'CONTINUE', /* required */
    LifecycleHookName: eventDetail['LifecycleHookName'], /* required */
    InstanceId: eventDetail['EC2InstanceId'],
    LifecycleActionToken: eventDetail['LifecycleActionToken']
  };
  var response;
  const command = new CompleteLifecycleActionCommand(params);
  try {
    var data = await autoscaling.send(command);
    console.log(data); // successful response
    response = {
      statusCode: 200,
      body: JSON.stringify('SUCCESS'),
    };
  } catch (err) {
    console.log(err, err.stack); // an error occurred
    response = {
      statusCode: 500,
      body: JSON.stringify('ERROR'),
    };
  }
  return response;
};
```

Dieser Code protokolliert einfach das Ereignis, sodass Sie am Ende dieses Tutorials sehen können, dass ein Ereignis im CloudWatch Log-Log-Stream erscheint, das mit dieser Lambda-Funktion verknüpft ist.

10. Wählen Sie Bereitstellen.

Schritt 3: Erstellen Sie eine Regel EventBridge

Erstellen Sie eine EventBridge Regel, um Ihre Lambda-Funktion auszuführen. Weitere Informationen zur Verwendung von EventBridge finden Sie unter [Wird EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen verwendet.](#)

So erstellen Sie eine Regel mithilfe der Konsole

1. Öffnen Sie die [EventBridge-Konsole](#).
2. Wählen Sie im Navigationsbereich Regeln aus.
3. Wählen Sie Regel erstellen aus.
4. Zum Define rule detail (Festlegen der Regeldetails) gehen Sie folgendermaßen vor:
 - a. Geben Sie unter Name **LogAutoScalingEvent-rule** ein.
 - b. Bei Event bus (Ereignisbus) wählen Sie default (Standard) aus. Wenn ein AWS-Service in Ihrem Konto ein Ereignis generiert, wird es immer an den Standard-Event-Bus Ihres Kontos weitergeleitet.
 - c. Bei Regeltyp wählen Sie Regel mit einem Ereignismuster aus.
 - d. Wählen Sie Weiter aus.
5. Bei Build event pattern (Ereignis-Muster erstellen) gehen Sie wie folgt vor:
 - a. Wählen Sie als Eventquelle AWS Events oder EventBridge Partnerevents aus.
 - b. Scrollen Sie nach unten zu Ereignis-Muster und gehen Sie wie folgt vor:
 - c.
 - i. Wählen Sie für Ereignisquelle die Option AWS-Services aus.
 - ii. Für AWS-Service, wählen Sie Auto Scaling aus.
 - iii. Wählen Sie in Event Type (Ereignistyp) die Option Instance Launch and Terminate (Starten und Beenden von Instances) aus.
 - iv. Standardmäßig entspricht die Regel jedem Abskalierungs- oder Aufskalierungs-Ereignis. Um eine Regel zu erstellen, die Sie benachrichtigt, wenn ein Aufskalierungs-Ereignis vorliegt und eine Instance aufgrund eines Lebenszyklus-Hook in einen Wartezustand versetzt wird, wählen Sie Specific instance event(s) (Bestimmte Instance-Ereignisse) und wählen Sie EC2 Instance-launch Lifecycle Action (Lebenszyklusaktion beim Start von EC2-Instances) aus.

- v. Standardmäßig stimmt die Regel mit jeder Auto-Scaling-Gruppe in der Region überein. Damit die Regel mit einer bestimmten Auto-Scaling-Gruppe übereinstimmt, wählen Sie Specific group name(s) und wählen Sie dann die Gruppe aus.
 - vi. Wählen Sie Weiter aus.
6. Bei Select target(s) (Ziel(e) auswählen) gehen Sie wie folgt vor:
 - a. Für Target types (Zieltypen), wählen Sie AWS-Service aus.
 - b. Für Select a target (Ein Ziel auswählen), wählen die Option Lambda function (Lambda-Funktion) aus.
 - c. Wählen Sie für Funktion die Option LogAutoScalingEvent.
 - d. Klicken Sie zweimal auf Weiter.
7. Wählen Sie auf der Seite Überprüfen und erstellen die Option Regel erstellen aus.

Schritt 4: Hinzufügen eines Lebenszyklus-Hooks

In diesem Abschnitt fügen Sie einen Lebenszyklus-Hook hinzu, damit Lambda Ihre Funktion beim Start auf Instances ausführt.

So fügen Sie einen Lebenszyklus-Hook hinzu

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe. Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.
3. Wählen Sie im unteren Bereich auf der Registerkarte Instance management (Instance-Verwaltung) unter Lebenszyklus-Hooks die Option Create Lebenszyklus hook (Lebenszyklus-Hook erstellen) aus.
4. Gehen Sie wie folgt vor, um einen Lebenszyklus-Hook zum Aufskalieren (Start von Instances) zu definieren:
 - a. Geben Sie für Lebenszyklus-Hooks den Wert **LogAutoScalingEvent-hook** ein.
 - b. Wählen Sie bei Lifecycle Transition (Lebenszykluswechsel) die Option Instance launch (Instance-Start) aus.
 - c. Für Heartbeat-Zeitüberschreitung geben Sie den Wert **300** für die Anzahl an Sekunden ein, um auf einen Rückruf von Ihrer Lambda-Funktion zu warten.

- d. Für Standardergebnis wählen Sie **ABBRECHEN** aus. Dies bedeutet, dass die Auto-Scaling-Gruppe eine neue Instance beendet, wenn die Zeitüberschreitung des Hook erreicht ist, ohne einen Rückruf von Ihrer Lambda-Funktion erhalten zu haben.
 - e. (Optional) Lassen Sie Benachrichtigungs-Metadaten leer. Die Ereignisdaten, an die wir übergeben, EventBridge enthalten alle notwendigen Informationen, um die Lambda-Funktion aufzurufen.
5. Wählen Sie **Erstellen** aus.

Schritt 5: Testen und Prüfen des Ereignisses

Um das Ereignis zu testen, aktualisieren Sie die Auto-Scaling-Gruppe, indem Sie die gewünschte Kapazität der Auto-Scaling-Gruppe um 1 erhöhen. Ihre Lambda-Funktion wird innerhalb weniger Sekunden nach der Erhöhung der gewünschten Kapazität aufgerufen.

So erhöhen Sie die Größe der Auto-Scaling-Gruppe

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe, um Details in einem unteren Bereich anzuzeigen und weiterhin die oberen Zeilen des oberen Bereichs anzuzeigen.
3. Wählen Sie im unteren Bereich auf der Registerkarte Details die Option **Gruppendetails, Bearbeiten** aus.
4. Erhöhen Sie für **Desired capacity** (Gewünschte Kapazität den aktuellen Wert um 1.
5. Wählen Sie **Aktualisieren** aus. Während eine Instance gestartet wird, zeigt die Spalte **Status** den **Status Updating capacity** (Kapazität aktualisieren) an.

Nachdem Sie die gewünschte Kapazität erhöht haben, können Sie prüfen, ob die Lambda-Funktion aufgerufen wurde.

Anzeigen der Ausgabe aus der Lambda-Funktion

1. Öffnen Sie die [Seite Protokollgruppen](#) der CloudWatch Konsole.
2. Wählen Sie den Namen der Protokollgruppe für Ihre Lambda-Funktion aus (`/aws/lambda/LogAutoScalingEvent`).
3. Wählen Sie den Namen des Protokoll-Streams aus, um die von der Funktion für die Lebenszyklus-Aktion bereitgestellten Daten anzuzeigen.

Als Nächstes können Sie anhand der Beschreibung der Skalierungsaktivitäten prüfen, ob die Instance erfolgreich gestartet wurde.

Ansehen der Skalierungsaktivität

1. Wählen Sie auf der Seite Auto-Scaling-Gruppen Ihre Gruppe aus.
2. Auf der Registerkarte Activity (Aktivität) wird unter Activity history (Aktivitätsverlauf) in der Spalte Status angezeigt, ob Ihre Auto-Scaling-Gruppe Instances erfolgreich gestartet hat.
 - Wenn die Aktion erfolgreich war, hat die Skalierungsaktivität den Status „Erfolgreich“.
 - Wenn es fehlgeschlagen ist, sehen Sie nach einigen Minuten eine Skalierungsaktivität mit dem Status „Abgebrochen“ und die Statusmeldung „Instance konnte nicht abgeschlossen werden: Lebenszyklusaktion des Benutzers: Lebenszyklusaktion mit Token E85EB647-4FE0-4909-B341-A6C42Beispiel wurde abgebrochen: Lebenszyklusaktion mit ABBRUCH-Ergebnis abgeschlossen“.

So verkleinern Sie die Auto-Scaling-Gruppe

Wenn Sie die zusätzliche Instance, die Sie für diesen Test gestartet haben, nicht benötigen, können Sie die Registerkarte Details öffnen und Desired capacity (Gewünschte Kapazität) um 1 reduzieren.

Schritt 6: Bereinigen

Wenn Sie mit den Ressourcen gearbeitet haben, die Sie speziell für dieses Tutorial erstellt haben, führen Sie die folgenden Schritte aus, um sie zu löschen.

So löschen Sie den Lebenszyklus-Hook

1. Öffnen Sie die [Seite Auto-Scaling-Gruppen](#) in der Amazon-EC2-Konsole.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.
3. Wählen Sie auf der Registerkarte Instance management (Instance-Verwaltung) unter Lebenszyklus-Hooks den Lebenszyklus (LogAutoScalingEvent-hook) aus.
4. Wählen Sie Actions (Aktionen), Delete (Löschen) aus.
5. Um dies zu bestätigen, wählen Sie erneut Delete (Löschen) aus.

Um die EventBridge Amazon-Regel zu löschen

1. Öffnen Sie die [Seite Regeln](#) in der EventBridge Amazon-Konsole.

2. Wählen Sie in Event bus (Ereignisbus) den Ereignisbus aus, der der Regel zugeordnet ist (Default).
3. Aktivieren Sie das Kontrollkästchen neben Ihrer Regel (LogAutoScalingEvent-rule).
4. Wählen Sie Löschen aus.
5. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie den Namen der Anwendung ein, und wählen Sie dann Delete aus.

Wenn Sie mit der Beispielfunktion fertig sind, löschen Sie sie. Sie können auch die Protokollgruppe löschen, welche die Protokolle der Funktion speichert, sowie die von Ihnen erstellte Ausführungsrolle und Berechtigungsrichtlinie.

So löschen Sie eine Lambda-Funktion

1. Öffnen Sie die Seite [Funktionen](#) der Lambda-Konsole.
2. Wählen Sie die Funktion (LogAutoScalingEvent) aus.
3. Wählen Sie Aktionen, Löschen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **delete**um die angegebene Funktion zu löschen und wählen Sie dann Löschen.

So löschen Sie die Protokollgruppe

1. Öffnen [Sie die Seite Protokollgruppen](#) der CloudWatch Konsole.
2. Wählen Sie die Protokollgruppe der Funktion (/aws/lambda/LogAutoScalingEvent).
3. Wählen Sie Actions (Aktionen), Delete log group(s) (Protokollgruppe(n) löschen) aus.
4. Wählen Sie im Dialogfeld Delete log group(s) (Protokollgruppe(n) löschen) die Option Delete (Löschen) aus.

So löschen Sie die Ausführungsrolle

1. Öffnen Sie die Seite [Roles \(Rollen\)](#) in der IAM-Konsole.
2. Wählen Sie die Rolle der Funktion (LogAutoScalingEvent-role).
3. Wählen Sie Löschen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie den Namen der Rolle ein und wählen Sie dann Delete (Löschen).

Löschen der IAM-Richtlinie

1. Öffnen Sie die Seite [Richtlinien](#) in der IAM-Konsole.
2. Wählen Sie die Richtlinie aus, die Sie erstellt haben (LogAutoScalingEvent-policy).
3. Wählen Sie Aktionen, Löschen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie den Namen der Richtlinie ein und wählen Sie dann Delete (Löschen).

Zugehörige Ressourcen

Die folgenden verwandten Themen können hilfreich sein, wenn Sie EventBridge Regeln erstellen, die auf Ereignissen basieren, die den Instances in Ihrer Auto Scaling Scaling-Gruppe passieren.

- [Wird EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen verwendet](#). In diesem Abschnitt finden Sie Beispiele für Ereignisse für andere Anwendungsfälle, einschließlich Ereignisse für die Skalierung.
- [Lebenszyklus-Hooks hinzufügen \(Konsole\)](#). Diese Prozedur zeigt Ihnen, wie Sie Lebenszyklus-Hooks sowohl für Aufskalieren (Start von Instances) als auch Abskalieren (Beendigung von Instances oder Rückkehr zu einem warmen Pool) hinzufügen können.

Ein Tutorial, das Ihnen zeigt, wie Sie den Instance Metadata Service (IMDS) verwenden, um eine Aktion innerhalb der Instance selbst aufzurufen, finden Sie unter [Tutorial: Verwenden Sie Datenskript- und Instance-Metadaten, um den Lebenszyklusstatus abzurufen](#).

Reduzieren Sie die Latenz für Anwendungen mit langen Startzeiten mithilfe warmer Pools

Ein Warm-Pool bietet Ihnen die Möglichkeit, die Latenz für Anwendungen mit außergewöhnlich langen Startzeiten zu verringern, z. B. weil Instances große Datenmengen auf die Festplatte schreiben müssen. Mit Warm-Pools müssen Sie Ihre Auto-Scaling-Gruppen nicht mehr übermäßig bereitstellen, um die Latenz zu verwalten, damit die Anwendungsleistung verbessert werden kann. Weitere Informationen finden Sie im Blog-Beitrag [Scaling your applications faster with EC2 Auto Scaling Warm Pools](#).

Important

Das Erstellen eines Warm-Pools, wenn dieser nicht erforderlich ist, kann zu unnötigen Kosten führen. Wenn Ihre erste Startzeit keine merklichen Latenzprobleme für Ihre Anwendung verursacht, benötigen Sie wahrscheinlich keinen Warm-Pool.

Topics

- [Schlüsselkonzepte](#)
- [Voraussetzungen](#)
- [Aktualisieren der Instances in einem warmen Pool](#)
- [Zugehörige Ressourcen](#)
- [Einschränkungen](#)
- [Verwenden Sie Lifecycle-Hooks mit einem warmen Pool in der Auto Scaling Scaling-Gruppe](#)
- [Erstellen eines warmen Pools für eine Auto-Scaling-Gruppe](#)
- [Anzeigen des Status der Zustandsprüfung und dem Grund für Zustandsprüfungsfehler](#)
- [Beispiele für die Erstellung und Verwaltung warmer Pools mit dem AWS CLI](#)

Schlüsselkonzepte

Bevor Sie beginnen, sollten Sie sich mit folgenden Kernkonzepten vertraut machen:

Warm Pool

Ein Warm Pool ist ein Pool vorinitialisierter EC2-Instances, der sich neben einer Auto-Scaling-Gruppe befindet. Jedes Mal, wenn Ihre Anwendung skaliert werden muss, kann die Auto-Scaling-Gruppe auf den Warm-Pool zurückgreifen, um die neue gewünschte Kapazität zu erfüllen. Er stellt sicher, dass Instances den Anwendungsdatenverkehr schnell bedienen können, wodurch die Reaktion auf eine Aufskalierung beschleunigt wird. Wenn Instances den Warm-Pool verlassen, zählen sie zur gewünschten Kapazität der Gruppe. Dies wird als Warmstart bezeichnet.

Während sich Instances im Warm Pool befinden, skalieren Ihre Skalierungsrichtlinien nur, wenn der Metrikwert von Instances, die sich im InService-Zustand befinden, größer ist als der hohe Alarmschwellenwert der Skalierungsrichtlinie (welcher der Zielauslastung einer Skalierungsrichtlinie für die Zielverfolgung entspricht).

Warm-Pool-Größe

Die Größe des Warm Pool wird standardmäßig als Differenz zwischen zwei Zahlen berechnet: die maximale Kapazität der Auto-Scaling-Gruppe und die gewünschte Kapazität. Wenn beispielsweise die gewünschte Kapazität Ihrer Auto-Scaling-Gruppe sechs ist und die maximale Kapazität zehn beträgt, beträgt die Größe Ihres Warm-Pools vier, wenn Sie den Warm-Pool zum ersten Mal einrichten und der Pool initialisiert wird.

Um die maximale Kapazität des warmen Pools separat anzugeben, verwenden Sie die benutzerdefinierte Spezifikationsoption (`MaxGroupPreparedCapacity`) und legen Sie dafür einen benutzerdefinierten Wert fest, der größer ist als die aktuelle Kapazität der Gruppe. Wenn Sie einen benutzerdefinierten Wert angeben, wird die Größe des warmen Pools als Differenz zwischen dem benutzerdefinierten Wert und der aktuell gewünschten Kapazität der Gruppe berechnet. Wenn die gewünschte Kapazität Ihrer Auto Scaling Scaling-Gruppe beispielsweise 6 ist, wenn die maximale Kapazität 20 ist und wenn der benutzerdefinierte Wert 8 ist, wird die Größe Ihres warmen Pools 2 sein, wenn Sie den warmen Pool zum ersten Mal einrichten und der Pool initialisiert wird.

Möglicherweise müssen Sie die Option benutzerdefinierte Spezifikation (`MaxGroupPreparedCapacity`) nur verwenden, wenn Sie mit großen Auto Scaling Scaling-Gruppen arbeiten, um die Kostenvorteile eines warmen Pools zu nutzen. So könnte zum Beispiel eine Auto-Scaling-Gruppe mit 1 000 Instances, einer maximalen Kapazität von 1 500 (um zusätzliche Kapazität für Notfall-Datenverkehrsspitzen bereitzustellen) und einem Warm Pool mit 100 Instances Ihre Ziele besser erreichen als mit einem Warm Pool mit 500 Instances.

Mindestanzahl des Warm-Pools

Erwägen Sie, die Einstellung für die Mindestgröße (`MinSize`) zu verwenden, um die Mindestanzahl der Instanzen, die im warmen Pool verwaltet werden sollen, statisch festzulegen. Standardmäßig ist keine Mindestgröße festgelegt. Die `MinSize` Einstellung ist nützlich, wenn Sie angeben `MaxGroupPreparedCapacity`, dass eine Mindestanzahl von Instanzen im warmen Pool auch dann beibehalten werden soll, wenn die gewünschte Kapazität der Auto Scaling Scaling-Gruppe höher als die `MaxGroupPreparedCapacity` ist.

Status der Warm-Pool-Instance

Sie können Instances im Warm Pool in einem von drei Status belassen: `Stopped`, `Running` oder `Hibernated`. Wenn Instances in einem `Stopped`-Zustand gelassen werden, können Kosten minimiert werden. Bei gestoppten Instances zahlen Sie nur für die Volumes, die Sie verwenden, und die Elastic-IP-Adressen, die den Instances angefügt sind.

Alternativ können Sie Instances auch in einem `Hibernated`-Status belassen, um Instances zu stoppen, ohne ihren Speicherinhalt (RAM) zu löschen. Wenn eine Instance in den Ruhezustand versetzt wird, signalisiert dies dem Betriebssystem, den Inhalt Ihres Arbeitsspeichers auf Ihrem Amazon-EBS-Root-Volume zu speichern. Wenn die Instance wieder gestartet wird, wird das Stamm-Volume im vorherigen Status wiederhergestellt und der RAM-Inhalt wird neu geladen. Während sich die Instances im Ruhezustand befinden, zahlen Sie nur für die EBS-Volumes, einschließlich des Speichers für die RAM-Inhalte und die mit den Instances verbundenen Elastic-IP-Adressen.

Instances in einem `Running` Zustand innerhalb des Warm-Pools zu halten, ist ebenfalls möglich, wird aber dringend abgeraten, um unnötige Gebühren zu vermeiden. Wenn Instances gestoppt oder in den Ruhezustand versetzt werden, sparen Sie die Kosten für die Instances selbst. Sie zahlen für die Instances nur, wenn sie ausgeführt werden.

Lebenszyklus-Hooks

Mit [Lebenszyklus-Hooks](#) können Sie Instances in einen Wartestatus versetzen, damit Sie benutzerdefinierte Aktionen für die Instances ausführen können. Benutzerdefinierte Aktionen werden beim Start der Instances oder vor dem Beenden ausgeführt.

In einer Warm-Pool-Konfiguration verzögern Lebenszyklus-Hooks, dass Instances gestoppt oder in den Ruhezustand versetzt werden und während des Aufskalierens in Betrieb genommen werden, bis sie die Initialisierung abgeschlossen haben. Wenn Sie Ihrer Auto-Scaling-Gruppe einen Warm Pool ohne Lebenszyklus-Hook hinzufügen, können Instances, deren Initialisierung lange dauert, gestoppt oder in den Ruhezustand versetzt und dann während der Aufskalierung in Betrieb genommen werden, bevor sie fertig sind.

Richtlinie für die Instance-Wiederverwendung

Standardmäßig beendet Amazon EC2 Auto Scaling Ihre Instances, wenn Ihre Auto-Scaling-Gruppe abskaliert. Dann startet die Lösung neue Instances im Warm Pool, um die beendeten Instances zu ersetzen.

Wenn Sie stattdessen Instances an den Warm Pool zurückgeben möchten, können Sie eine Richtlinie für die Instance-Wiederverwendung angeben. Auf diese Weise können Sie Instances wiederverwenden, die bereits für die Bereitstellung des Anwendungsdatenverkehrs konfiguriert sind. Um sicherzustellen, dass Ihr Warm Pool nicht überdimensioniert wird, kann Amazon EC2 Auto Scaling Instances im Warm Pool beenden, um seine Größe zu reduzieren, wenn sie die ursprünglichen Einstellungen übersteigt. Wenn Instances im Warm Pool beendet werden, wird

die [Standardbeendigungsrichtlinie](#) verwendet, um auszuwählen, welche Instances zuerst beendet werden sollen.

Important

Wenn Sie Instances beim Abskalieren in den Ruhezustand versetzen möchten und Instances in der Auto-Scaling-Gruppe vorhanden sind, müssen diese die Anforderungen für den Instance-Ruhezustand erfüllen. Wenn dies nicht der Fall ist, werden Instances gestoppt, und nicht in den Ruhezustand versetzt, wenn sie in den Warm Pool zurückkehren.

Note

Derzeit können Sie eine Richtlinie für die Instance-Wiederverwendung nur mithilfe der AWS CLI oder eines SDK angeben. Diese Funktion ist in der Konsole nicht verfügbar.

Voraussetzungen

Bevor Sie einen warmen Pool für Ihre Auto-Scaling-Gruppe erstellen, entscheiden Sie, wie Sie Lebenszyklus-Hooks verwenden, um neue Instances mit einem geeigneten Anfangszustand zu initialisieren.

Um benutzerdefinierte Aktionen für Instances auszuführen, während sich diese aufgrund eines Lebenszyklus-Hooks im Wartestatus befinden, haben Sie zwei Möglichkeiten:

- Für einfache Szenarien, in denen Sie beim Start Befehle für Ihre Instances ausführen möchten, können Sie ein Benutzerdatenskript einbeziehen, wenn Sie eine Startvorlage erstellen oder eine Konfiguration für Ihre Auto-Scaling-Gruppe starten. Benutzerdatenskripts sind normale Shell-Skripts oder cloud-init Direktiven, die cloud-init beim Start Ihrer Instances ausgeführt werden. Das Skript kann auch steuern, wann Ihre Instances in den nächsten Status übergehen, indem Sie die ID der Instance verwenden, auf der sie ausgeführt wird. Wenn Sie nicht bereits dabei sind, aktualisieren Sie das Skript, sodass es die Instance-ID der Instance aus den Instance-Metadaten abrufen. Weitere Informationen finden Sie unter [Zugriff auf Instance-Metadaten](#) im Amazon EC2 EC2-Benutzerhandbuch.

i Tip

Um Benutzerdatenskripte beim Neustart einer Instance auszuführen, müssen die Benutzerdaten im mehrteiligen MIME-Format vorliegen und Folgendes im Abschnitt `#cloud-config` der Benutzerdaten angeben:

```
#cloud-config
cloud_final_modules:
  - [scripts-user, always]
```

- Für fortgeschrittene Szenarien, in denen Sie einen Dienst benötigen, AWS Lambda um beispielsweise etwas zu tun, wenn Instances den warmen Pool betreten oder verlassen, können Sie einen Lifecycle-Hook für Ihre Auto Scaling Scaling-Gruppe erstellen und den Zieldienst so konfigurieren, dass er benutzerdefinierte Aktionen auf der Grundlage von Lebenszyklusbenachrichtigungen ausführt. Weitere Informationen finden Sie unter [Unterstützte Benachrichtigungsziele](#).

Instances auf Ruhezustand vorbereiten

Um Auto Scaling Scaling-Instances für die Verwendung des Hibernated Pool-Status vorzubereiten, erstellen Sie eine neue Startvorlage oder Startkonfiguration, die korrekt eingerichtet ist, um den Instance-Ruhezustand zu unterstützen, wie im Thema [Voraussetzungen für den Ruhezustand](#) im Amazon EC2 EC2-Benutzerhandbuch beschrieben. Ordnen Sie dann die neue Startvorlage oder Startkonfiguration der Auto-Scaling-Gruppe zu und starten Sie eine Instance-Aktualisierung, um die Instances zu ersetzen, die einer vorherigen Startvorlage oder Startkonfiguration zugeordnet sind. Weitere Informationen finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).

Aktualisieren der Instances in einem warmen Pool

Um die Instances in einem warmen Pool zu aktualisieren, erstellen Sie eine neue Startvorlage oder Startkonfiguration und verknüpfen sie mit der Auto-Scaling-Gruppe. Alle neuen Instances werden mithilfe des neuen AMI und anderer Updates gestartet, die in der Startvorlage oder Startkonfiguration angegeben sind, bestehende Instances sind jedoch nicht davon betroffen.

Um das Starten von warmen Ersatz-Pool-Instances zu erzwingen, die die neue Startvorlage oder Startkonfiguration verwenden, können Sie eine Instance-Aktualisierung starten, um eine fortlaufende

Aktualisierung Ihrer Gruppe durchzuführen. Eine Instance-Aktualisierung ersetzt zuerst InService-Instances. Dann ersetzt sie Instances im Warm-Pool. Weitere Informationen finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).

Zugehörige Ressourcen

In unserem [GitHubRepository](#) finden Sie Beispiele für Lifecycle-Hooks für warme Pools.

Einschränkungen

- Warm-Pool-Einschränkungen für eine Auto Scaling Scaling-Gruppe mit gemischten Instance-Typen:
 - Warme Pools werden bei gewichteten gemischten Instanzgruppen nicht unterstützt. Wenn Ihre Auto Scaling Scaling-Gruppe Instance-Gewichtung verwendet, können Sie keinen warmen Pool hinzufügen.
 - Warme Pools unterstützen keine Spot-Instances innerhalb gemischter Instanzgruppen. Ihre Richtlinie für gemischte Instanzen muss nur für On-Demand-Instances konfiguriert werden, wenn Sie warme Pools verwenden.
 - Wenn Sie warme Pools mit gemischten Instanzgruppen im Ruhezustand verwenden, müssen Sie die Konfiguration `HibernationOptions` in Ihrer Startvorlage vornehmen.
- Amazon EC2 Auto Scaling kann eine Instance nur dann in einen Stopped- oder Hibernated-Status verschieben, wenn es ein Amazon-EBS-Volume als Stammgerät hat. Instances, die Instance-Speicher als Stammgerät verwenden, können nicht beendet oder in den Ruhezustand versetzt werden.
- Amazon EC2 Auto Scaling kann eine Instance nur dann in einen Hibernated Zustand versetzen, wenn sie alle im Thema [Voraussetzungen für den Ruhezustand](#) im Amazon EC2 EC2-Benutzerhandbuch aufgeführten Anforderungen erfüllt.
- Wenn Ihr Warm-Pool bei einem Scale-Out-Ereignis erschöpft ist, werden Instances direkt in der Auto-Scaling-Gruppe (ein Kaltstart) starten. Es kann auch zu einem Kaltstart kommen, wenn eine Availability Zone nicht genügend Kapazität hat.
- Wenn eine Instance innerhalb des Warmpools während des Startvorgangs auf ein Problem stößt, das sie daran hindert, den InService Status zu erreichen, wird die Instance als fehlgeschlagener Start betrachtet und beendet. Dies gilt unabhängig von der zugrunde liegenden Ursache, z. B. einem Fehler bei unzureichender Kapazität oder einem anderen Faktor.

- Wenn Sie versuchen, einen Warm-Pool mit einer von Amazon Elastic Kubernetes Service (Amazon EKS) verwalteten Knotengruppe zu verwenden, registrieren sich Instances, die noch initialisiert werden, möglicherweise bei Ihrem Amazon-EKS-Cluster. Infolgedessen kann der Cluster Jobs für eine Instance planen, da er sich darauf vorbereitet, gestoppt oder in den Ruhezustand versetzt zu werden.
- Wenn Sie versuchen, einen Warm-Pool mit einem Amazon ECS-Cluster zu verwenden, registrieren sich die Instances eventuell beim Cluster, bevor sie ihre Initialisierung abgeschlossen haben. Um dieses Problem zu lösen, müssen Sie eine Startvorlage oder Startkonfiguration konfigurieren, die eine spezielle Agentenkonfigurationsvariable in den Benutzerdaten enthält. Weitere Informationen finden Sie unter [Verwendung eines Warm-Pools für Ihre Auto-Scaling-Gruppe](#) im Amazon Elastic Container Service-Entwicklerhandbuch.

Verwenden Sie Lifecycle-Hooks mit einem warmen Pool in der Auto Scaling Scaling-Gruppe

Instances in einem Warm Pool verwalten ihren eigenen, unabhängigen Lebenszyklus, um Ihnen bei der Erstellung der entsprechenden benutzerdefinierten Aktion für jeden Übergang zu helfen. Dieser Lebenszyklus soll Ihnen helfen, Aktionen in einem Zielservice (z. B. einer Lambda-Funktion) aufzurufen, während eine Instance noch initialisiert wird und bevor sie in Betrieb genommen wird.

Note

Die API-Vorgänge, die Sie zum Hinzufügen und Verwalten von Lebenszyklus-Hooks und zum Abschließen von Lebenszyklusaktionen verwenden, werden nicht geändert. Nur der Instance-Lebenszyklus wird geändert.

Weitere Informationen über das Hinzufügen von Lebenszyklus-Hooks finden Sie unter [Fügen Sie Lifecycle-Hooks zu Ihrer Auto Scaling Scaling-Gruppe hinzu](#). Weitere Informationen über das Abschließen einer Lebenszyklus-Aktion finden Sie unter [Eine Lebenszyklusaktion in einer Auto Scaling Scaling-Gruppe abschließen](#).

Für Instances, die in den Warm Pool aufgenommen werden, benötigen Sie möglicherweise aus einem der folgenden Gründe einen Lebenszyklus-Hook:

- Sie möchten EC2-Instances von einem AMI aus starten, dessen Initialisierung lange dauert.
- Sie möchten Benutzerdatenskripte ausführen, um die EC2-Instances zu laden.

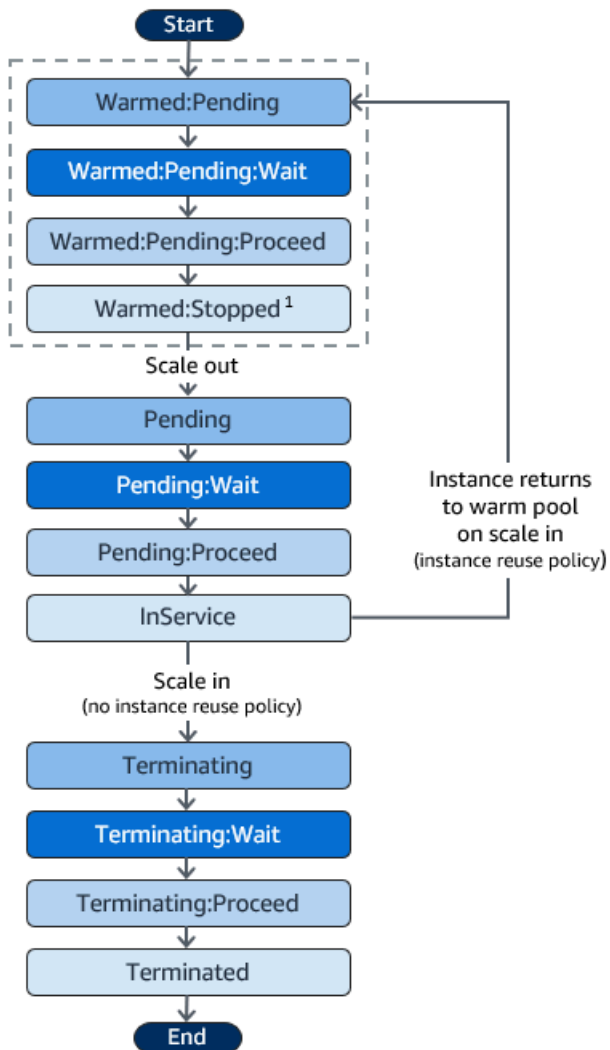
Für Instances, die den Warm Pool verlassen, benötigen Sie möglicherweise aus einem der folgenden Gründe einen Lebenszyklus-Hook:

- Sie können etwas mehr Zeit gebrauchen, um EC2-Instances auf die Verwendung vorzubereiten. Sie haben möglicherweise Services, die gestartet werden müssen, wenn eine Instance neu gestartet wird, bevor Ihre Anwendung ordnungsgemäß funktionieren kann.
- Sie möchten Cache-Daten vorab ausfüllen, um sicherzustellen, dass ein neuer Server nicht mit einem leeren Cache gestartet wird.
- Sie möchten neue Instances als verwaltete Instances mit Ihrem Konfigurationsverwaltungsservice registrieren.

Lebenszyklusstatusübergänge für Instances in einem Warm Pool

Eine Instance mit automatischer Skalierung kann im Verlauf ihres Lebenszyklus in verschiedene Status übergehen.

Im folgenden Diagramm wird der Übergang zwischen Status für automatische Skalierung veranschaulicht, wenn Sie einen Warm Pool verwenden:



¹ Dieser Status variiert je nach Einstellung des Pool-Status des Warm Pools. Wenn der Pool-Status auf Running gesetzt ist, ist dieser Status stattdessen Warmed:Running. Wenn der Pool-Status auf Hibernated gesetzt ist, ist dieser Status stattdessen Warmed:Hibernated.

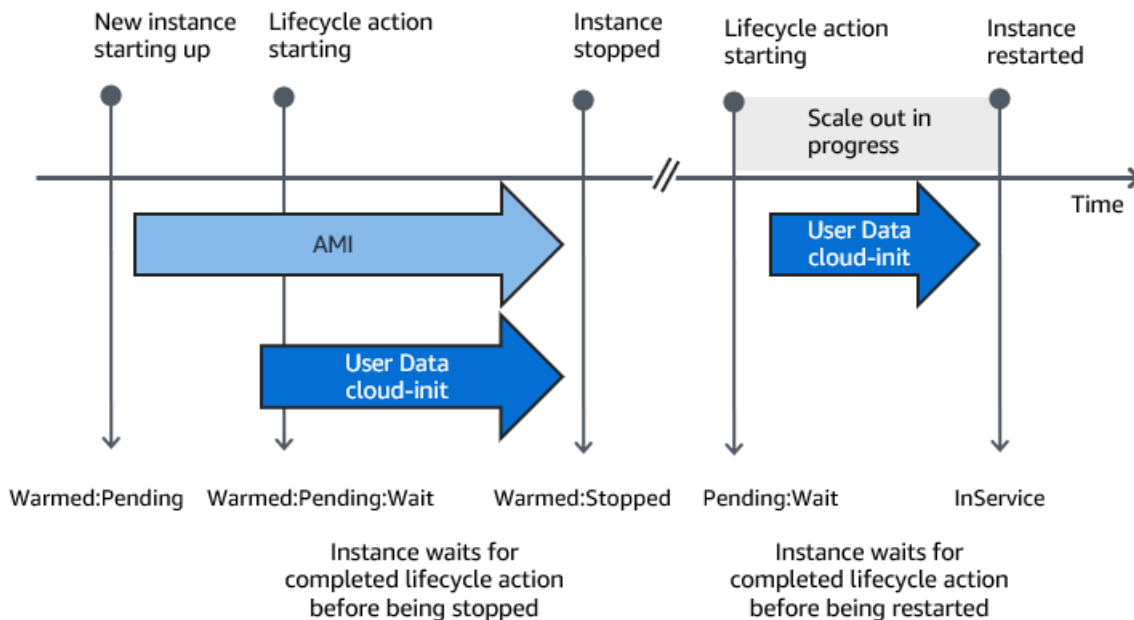
Berücksichtigen Sie beim Hinzufügen von Lebenszyklus-Hooks Folgendes:

- Wenn ein Lebenszyklus-Hook für die `autoscaling:EC2_INSTANCE_LAUNCHING`-Lebenszyklusaktion konfiguriert ist, wird eine neu gestartete Instance zunächst angehalten, um eine benutzerdefinierte Aktion auszuführen, wenn sie den `Warmed:Pending:Wait`-Zustand erreicht, und dann erneut, wenn die Instance neu gestartet wird und den `Pending:Wait`-Zustand erreicht.
- Wenn ein Lebenszyklus-Hook für die `EC2_INSTANCE_TERMINATING`-Lebenszyklusaktion konfiguriert ist, wird eine beendende Instance angehalten, um eine benutzerdefinierte Aktion auszuführen, wenn sie den `Terminating:Wait`-Zustand erreicht. Wenn Sie jedoch eine

Richtlinie für die Wiederverwendung von Instances so festlegen, dass Instances bei der Abwärtsskalierung in den Warm-Pool zurückgeführt werden, anstatt sie zu beenden, dann wird eine Instanz, die in den Warm-Pool zurückkehrt, angehalten, um eine benutzerdefinierte Aktion im `Warmed:Pending:Wait`-Zustand für die `EC2_INSTANCE_TERMINATING`-Lebenszyklusaktion durchzuführen.

- Wenn die Nachfrage nach Ihrer Anwendung den Warm Pool leert, kann Amazon EC2 Auto Scaling Instances direkt in der Auto-Scaling-Gruppe starten, sofern die Gruppe ihre maximale Kapazität noch nicht erreicht hat. Wenn die Instances direkt in der Gruppe gestartet werden, werden sie nur angehalten, um eine benutzerdefinierte Aktion im `Pending:Wait`-Zustand auszuführen.
- Um zu steuern, wie lange eine Instance in einem Wartezustand verbleibt, bevor sie in den nächsten Status übergeht, konfigurieren Sie Ihre benutzerdefinierte Aktion so, dass sie den `complete-lifecycle-action`-Befehl verwendet. Mit Lebenszyklus-Hooks bleiben Instances in einem Wartezustand, bis Sie Amazon EC2 Auto Scaling benachrichtigen, dass die angegebene Lebenszyklusaktion abgeschlossen ist, oder bis eine Zeitüberschreitung auftritt (standardmäßig eine Stunde).

Im Folgenden wird der Ablauf für ein Aufskalierungsereignis zusammengefasst.



Wenn Instances einen Wartezustand erreichen, sendet Amazon EC2 Auto Scaling eine Benachrichtigung. Beispiele für diese Benachrichtigungen finden Sie im EventBridge Abschnitt dieses Handbuchs. Weitere Informationen finden Sie unter [Beispielereignisse und -muster in einem warmen Pool](#).

Unterstützte Benachrichtigungsziele

Amazon EC2 Auto Scaling bietet Unterstützung bei der Definition einer der folgenden Optionen als Benachrichtigungsziele für Lebenszyklusbenachrichtigungen:

- EventBridge Regeln
- Amazon-SNS-Themen
- Amazon-SQS-Warteschlangen
- AWS Lambda Funktionen

Important

Wenn Sie in Ihrer Startvorlage oder Startkonfiguration ein Benutzerdatenskript (Cloud-Init) haben, das Ihre Instances beim Start konfiguriert, benötigen Sie keine Benachrichtigungen, um benutzerdefinierte Aktionen für Instances auszuführen, die gerade gestartet oder neu gestartet werden.

Die folgenden Abschnitte enthalten Links zur Dokumentation, in der beschrieben wird, wie Benachrichtigungsziele konfiguriert werden:

EventBridge Regeln — Um Code auszuführen, wenn Amazon EC2 Auto Scaling eine Instance in einen Wartestatus versetzt, können Sie eine EventBridge Regel erstellen und eine Lambda-Funktion als Ziel angeben. Um verschiedene Lambda-Funktionen basierend auf verschiedenen Lebenszyklusbenachrichtigungen aufzurufen, können Sie mehrere Regeln erstellen und jede Regel einem bestimmten Ereignismuster und einer Lambda-Funktion zuordnen. Weitere Informationen finden Sie unter [Erstellen Sie EventBridge Regeln für Ereignisse im warmen Pool](#).

Amazon SNS SNS-Themen — Um eine Benachrichtigung zu erhalten, wenn eine Instance in einen Wartestatus versetzt wird, erstellen Sie ein Amazon SNS SNS-Thema und richten dann die Amazon SNS SNS-Nachrichtenfilterung so ein, dass Lebenszyklusbenachrichtigungen je nach Nachrichtenattribut unterschiedlich zugestellt werden. Weitere Informationen finden Sie unter [Benachrichtigungen über Amazon SNS erhalten](#).

Amazon SQS SQS-Warteschlangen — Um einen Lieferpunkt für Lebenszyklusbenachrichtigungen einzurichten, an dem ein relevanter Verbraucher sie abholen und verarbeiten kann, können Sie eine Amazon SQS SQS-Warteschlange und einen Warteschlangenverbraucher erstellen, der

Nachrichten aus der SQS-Warteschlange verarbeitet. Wenn der Warteschlangenverbraucher Lebenszyklusbenachrichtigungen basierend auf einem Nachrichtenattribut unterschiedlich verarbeiten soll, müssen Sie auch den Warteschlangenverbraucher so einrichten, dass er die Nachricht analysiert und dann auf die Nachricht reagiert, wenn ein bestimmtes Attribut dem gewünschten Wert entspricht. Weitere Informationen finden Sie unter [Benachrichtigungen über Amazon SQS erhalten](#).

AWS Lambda Funktionen — Um benutzerdefinierten Code auszuführen, wenn Amazon EC2 Auto Scaling eine Instance in einen Wartestatus versetzt, können Sie eine Lambda-Funktion als Benachrichtigungsziel angeben. Die Lambda-Funktion wird mit Lebenszyklusbenachrichtigungsdaten aufgerufen, sodass Sie benutzerdefinierte Aktionen wie Instanzkonfiguration, Anwendungseinrichtung oder Integration mit anderen AWS Diensten durchführen können. Sie müssen die ressourcenbasierte Richtlinie der Lambda-Funktion so konfigurieren, dass die mit dem Service verknüpfte Auto Scaling Scrolling-Rolle die Funktion aufrufen kann. Weitere Informationen finden Sie unter [Benachrichtigungen AWS Lambda direkt weiterleiten](#).

Erstellen eines warmen Pools für eine Auto-Scaling-Gruppe

In diesem Thema wird beschrieben, wie Sie einen warmen Pool für Ihre Auto-Scaling-Gruppe erstellen.

Important

Bevor Sie fortfahren, sollten Sie die [Voraussetzungen](#) für die Erstellung eines warmen Pools erfüllen und bestätigen, dass Sie einen Lebenszyklus-Hook für Ihre Auto-Scaling-Gruppe erstellt haben.

Erstellen eines Warm Pool

Führen Sie die folgenden Schritte aus, um einen warmen Pool für Ihre Auto-Scaling-Gruppe zu erstellen.

So erstellen Sie einen Warm Pool (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie die Registerkarte Instance-Management aus.
4. Unter Warm-Pool wählen Sie Erstellen eines Warm-Pools aus.
5. Um einen Warm-Pool zu konfigurieren, gehen Sie folgendermaßen vor:
 - a. Für Warm-Pool-Instanzenstatus wählen Sie aus, in welchen Zustand Ihre Instances wechseln sollen, wenn sie in den Warm-Pool gelangen. Der Standardwert ist Stopped.
 - b. Für Mindestgröße des Warm-Pools geben Sie die Mindestanzahl der Instances ein, die im Warm-Pool verwaltet werden sollen.
 - c. Aktivieren Sie für die Wiederverwendung von Instances das Kontrollkästchen Skaliert wiederverwenden, damit Instances in der Auto Scaling-Gruppe in den warmen Pool zurückkehren können.
 - d. Wählen Sie für Größe des warmen Pools eine der verfügbaren Optionen aus:
 - Standardspezifikation: Die Größe des warmen Pools wird durch die Differenz zwischen der maximalen und der gewünschten Kapazität der Auto Scaling Scaling-Gruppe bestimmt. Diese Option optimiert die Verwaltung von warmen Pools. Nachdem Sie den warmen Pool erstellt haben, kann seine Größe einfach aktualisiert werden, indem Sie einfach die maximale Kapazität der Gruppe anpassen.
 - Benutzerdefinierte Spezifikation: Die Größe des warmen Pools wird durch die Differenz zwischen einem benutzerdefinierten Wert und der gewünschten Kapazität der Auto Scaling Scaling-Gruppe bestimmt. Diese Option bietet Ihnen die Flexibilität, die Größe Ihres warmen Pools unabhängig von der maximalen Kapazität der Gruppe zu verwalten.
6. Sehen Sie sich den Abschnitt Geschätzte Größe des warmen Pools anhand der aktuellen Einstellungen an, um zu überprüfen, ob die Standard- oder benutzerdefinierte Spezifikation für die Größe des Warmwasserbeckens gilt. Denken Sie daran, dass die Größe des warmen Pools von der gewünschten Kapazität der Auto Scaling Scaling-Gruppe abhängt, die sich ändert, wenn die Gruppe skaliert wird.
7. Wählen Sie Erstellen aus.

Auswahl des Instanztyps mit gemischten Instanzgruppen

Auto Scaling priorisiert Instanztypen, die sich bei Skalierungsereignissen bereits im warmen Pool befinden, wenn Ihre Gruppe mit einer Richtlinie für gemischte Instanzen konfiguriert ist. Verhalten beim Start:

1. Auto Scaling versucht, Instances mit Instance-Typen zu starten, die im warmen Pool verfügbar sind.
2. Wenn der Warmstart fehlschlägt, versucht Auto Scaling, einen Kaltstart unter Verwendung aller verbleibenden Instance-Typen in Ihrer Richtlinie für gemischte Instanzen zu starten.

Example

Beispiel

Wenn Sie Ihre Auto Scaling Scaling-Gruppe mit 10 Instance-Typen konfigurieren und Ihr Warm-Pool 6 dieser Instance-Typen enthält. Beim Scale-Out probiert Auto Scaling zunächst die 6 Instance-Typen aus dem warmen Pool aus. Wenn dies nicht gelingt, versucht Auto Scaling alle konfigurierten Instance-Typen per Kaltstart.

Auf diese Weise erhalten Sie, wenn möglich, Leistungsvorteile beim Warmpool und behalten gleichzeitig die Flexibilität Ihrer vollständigen Konfiguration mit gemischten Instanzen.

Löschen eines Warm-Pools

Wenn Sie eine DHCP-Optionsliste nicht mehr benötigen, können Sie sie mit dem folgenden Verfahren löschen.

So löschen Sie Ihren Warm Pool (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie die Registerkarte Instance-Management aus.
4. Wählen Sie für Warm pool (Warm Pool) Actions (Aktionen), Delete (Löschen) aus.
5. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Delete (Löschen).

Anzeigen des Status der Zustandsprüfung und dem Grund für Zustandsprüfungsfehler

Mithilfe von Zustandsprüfungen kann Amazon EC2 Auto Scaling feststellen, wann eine Instance fehlerhaft ist und beendet werden sollte. Für Warm-Pool-Instances, die in einem Stopped-

Zustand belassen werden, verwendet es die Kenntnisse, die Amazon EBS über eine Stopped-Instance hat, um fehlerhafte Instances zu identifizieren. Dies geschieht durch Aufrufen der DescribeVolumeStatus-API, um den Zustand des EBS-Volume zu ermitteln, das an die Instance angefügt ist. Für Warm-Pool-Instances, die in einem Running-Zustand belassen werden, stützt es sich auf EC2-Zustandsprüfungen, um den Instance-Zustand zu ermitteln. Zwar gibt es keine Zustandsprüfungsfrist für Warm-Pool-Instances, Amazon EC2 Auto Scaling startet die Prüfung der Instance-Integrität aber erst, wenn der Lebenszyklus-Hook abgeschlossen ist.

Wenn eine Instance als fehlerhaft erachtet wird, löscht Amazon EC2 Auto Scaling die fehlerhafte Instance automatisch und erstellt eine neue Instance, um sie zu ersetzen. Instances werden normalerweise innerhalb weniger Minuten nach erfolgter Zustandsprüfung beendet. Weitere Informationen finden Sie unter [Anzeigen des Grundes für Fehler bei Zustandsprüfung](#).

Benutzerdefinierte Zustandsprüfungen werden ebenfalls unterstützt. Dies kann hilfreich sein, wenn Sie über ein eigenes Zustandsprüfsystem verfügen, das den Zustand einer Instance erkennt und diese Informationen an Amazon EC2 Auto Scaling sendet. Weitere Informationen finden Sie unter [Richten Sie eine benutzerdefinierte Integritätsprüfung für Ihre Auto Scaling Scaling-Gruppe ein](#).

In der Amazon EC2 Auto Scaling-Konsole können Sie den Zustand (fehlerfrei oder fehlerhaft) Ihrer Warm-Pool-Instances einsehen. Sie können ihren Gesundheitsstatus auch mithilfe der AWS CLI oder einer der SDKs Optionen anzeigen.

So zeigen Sie den Zustand Ihrer Warm-Pool-Instances an (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Auf der Registerkarte Instance management (Instance-Verwaltung) wird unter Warm pool instances (Warm-Pool-Instances) in der Spalte Lifecycle (Lebenszyklus) der Zustand Ihrer Instances angezeigt.

Die Spalte Zustand zeigt die Bewertung an, die Amazon EC2 Auto Scaling über den Instance-Zustand gemacht hat.

Note

Neue Instances beginnen fehlerfrei. Bis der Lebenszyklus-Hook abgeschlossen ist, wird die Integrität einer Instance nicht überprüft.

So zeigen Sie den Grund für den Ausfall einer Zustandsprüfung an (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Auf der Registerkarte Activity (Aktivität) wird unter Activity history (Aktivitätsverlauf) in der Spalte Status angezeigt, ob Ihre Auto-Scaling-Gruppe Instances erfolgreich gestartet oder beendet hat.

Wenn es Instances fehlerhaft beendet hat, zeigt die Spalte Ursache das Datum und die Uhrzeit der Beendigung und den Grund für den Fehler der Zustandsprüfung an. Beispiel: „Bei 2021-04-01T21:48:35Z wurde eine Instance aufgrund eines Fehlers der EBS-Volumen-Zustandsprüfung außer Betrieb gesetzt“.

Anzeigen des Status Ihrer Warm-Pool-Instances (AWS CLI)

Zeigen Sie den warmen Pool für eine Auto Scaling Group mit dem folgenden [describe-warm-pool](#) Befehl an.

```
aws autoscaling describe-warm-pool --auto-scaling-group-name my-asg
```

Beispielausgabe.

```
{
  "WarmPoolConfiguration": {
    "MinSize": 0,
    "PoolState": "Stopped"
  },
  "Instances": [
    {
```

```

    "InstanceId": "i-0b5e5e7521cfaa46c",
    "InstanceType": "t2.micro",
    "AvailabilityZone": "us-west-2a",
    "LifecycleState": "Warmed:Stopped",
    "HealthStatus": "Healthy",
    "LaunchTemplate": {
      "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
      "LaunchTemplateName": "my-template-for-auto-scaling",
      "Version": "1"
    }
  },
  {
    "InstanceId": "i-0e21af9dcfb7aa6bf",
    "InstanceType": "t2.micro",
    "AvailabilityZone": "us-west-2a",
    "LifecycleState": "Warmed:Stopped",
    "HealthStatus": "Healthy",
    "LaunchTemplate": {
      "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
      "LaunchTemplateName": "my-template-for-auto-scaling",
      "Version": "1"
    }
  }
]
}

```

So zeigen Sie den Grund für den Ausfall einer Zustandsprüfung an (AWS CLI)

Verwenden Sie den folgenden [describe-scaling-activities](#)-Befehl.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

Nachfolgend finden Sie eine Beispielantwort, wobei `Description` angibt, dass Ihre Auto-Scaling-Gruppe eine Instance beendet hat, und `Cause` den Grund für den Fehler bei der Zustandsprüfung angibt.

Skalierungsaktivitäten werden nach Startzeit sortiert. Die noch laufenden Aktivitäten werden zuerst beschrieben.

```

{
  "Activities": [
    {

```

```
"ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
"AutoScalingGroupName": "my-asg",
"Description": "Terminating EC2 instance: i-04925c838b6438f14",
"Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in
response to EBS volume health check failure.",
"StartTime": "2021-04-01T21:48:35.859Z",
"EndTime": "2021-04-01T21:49:18Z",
"StatusCode": "Successful",
"Progress": 100,
"Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}\",
"AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
},
...
]
```

Beispiele für die Erstellung und Verwaltung warmer Pools mit dem AWS CLI

Sie können warme Pools mit dem AWS-Managementkonsole, AWS Command Line Interface (AWS CLI) oder erstellen und verwalten SDKs.

Die folgenden Beispiele zeigen, wie Sie Warm Pools mithilfe der AWS CLI erstellen und verwalten.

Inhalt

- [Beispiel 1: Instances im Zustand Stopped belassen](#)
- [Beispiel 2: Instances im Zustand Running belassen](#)
- [Beispiel 3: Instances im Zustand Hibernated belassen](#)
- [Beispiel 4: Instances beim Scale-In wieder in den Warm Pool verschieben](#)
- [Beispiel 5: Angeben der Mindestanzahl der Instances im Warm Pool](#)
- [Beispiel 6: Definieren Sie die Größe des warmen Pools mithilfe einer benutzerdefinierten Spezifikation](#)
- [Beispiel 7: Definieren einer absoluten Warm Pool-Größe](#)
- [Beispiel 8: Einen Warm Pool löschen](#)

Beispiel 1: Instances im Zustand **Stopped** belassen

Im folgenden [put-warm-pool](#)Beispiel wird ein warmer Pool erstellt, der Instanzen in einem bestimmten Stopped Zustand hält.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped
```

Beispiel 2: Instances im Zustand **Running** belassen

Im folgenden [put-warm-pool](#)Beispiel wird ein warmer Pool erstellt, der Instanzen in einem Running Zustand statt in einem Stopped Zustand hält.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Running
```

Beispiel 3: Instances im Zustand **Hibernated** belassen

Im folgenden [put-warm-pool](#)Beispiel wird ein warmer Pool erstellt, der Instanzen in einem Hibernated Status statt in einem Stopped Zustand hält. Auf diese Weise können Sie Instances stoppen, ohne ihren Speicherinhalt (RAM) zu löschen.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Hibernated
```

Beispiel 4: Instances beim Scale-In wieder in den Warm Pool verschieben

Im folgenden [put-warm-pool](#)Beispiel wird ein warmer Pool erstellt, der Instanzen in einem bestimmten Stopped Zustand hält und die `--instance-reuse-policy` Option enthält. Der Wert `'{"ReuseOnScaleIn": true}'` der Richtlinie für die Instance-Wiederverwendung weist Amazon EC2 Auto Scaling an, Instances an den Warm Pool zurückzugeben, wenn Ihre Auto-Scaling-Gruppe skaliert wird.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --instance-reuse-policy '{"ReuseOnScaleIn": true}'
```

Beispiel 5: Angeben der Mindestanzahl der Instances im Warm Pool

Im folgenden [put-warm-pool](#)Beispiel wird ein warmer Pool erstellt, der mindestens 4 Instanzen verwaltet, sodass mindestens 4 Instanzen zur Verfügung stehen, um Datenverkehrsspitzen zu bewältigen.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
  --pool-state Stopped --min-size 4
```

Beispiel 6: Definieren Sie die Größe des warmen Pools mithilfe einer benutzerdefinierten Spezifikation

Standardmäßig verwaltet Amazon EC2 Auto Scaling die Größe Ihres warmen Pools als Differenz zwischen der maximalen und der gewünschten Kapazität der Auto Scaling-Gruppe. Sie können die Größe des warmen Pools jedoch unabhängig von der maximalen Kapazität der Gruppe verwalten, indem Sie die `--max-group-prepared-capacity` Option verwenden.

Im folgenden [put-warm-pool](#)Beispiel wird ein warmer Pool erstellt und die maximale Anzahl von Instances festgelegt, die gleichzeitig sowohl im Warmpool als auch in der Auto Scaling Scaling-Gruppe existieren können. Wenn die Gruppe eine gewünschte Kapazität von 800 hat, hat der Warm-Pool zunächst eine Größe von 100, da er nach der Ausführung dieses Befehls initialisiert wird.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
  --pool-state Stopped --max-group-prepared-capacity 900
```

Um eine Mindestanzahl von Instances im Warm-Pool beizubehalten, fügen Sie die `--min-size`-Option mit dem Befehl wie folgt ein.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
  --pool-state Stopped --max-group-prepared-capacity 900 --min-size 25
```

Beispiel 7: Definieren einer absoluten Warm Pool-Größe

Wenn Sie die `--max-group-prepared-capacity`-und `--min-size`-Optionen auf den gleichen Wert setzen, wird der Warm Pool eine absolute Größe haben. Im folgenden [put-warm-pool](#)Beispiel wird ein Warmpool erstellt, der eine konstante Größe von 10 Instanzen beibehält.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
  --pool-state Stopped --min-size 10 --max-group-prepared-capacity 10
```

Beispiel 8: Einen Warm Pool löschen

Verwenden Sie den folgenden [delete-warm-pool](#) Befehl, um einen warmen Pool zu löschen.

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg
```

Wenn sich im warmen Pool Instanzen befinden oder Skalierungsaktivitäten im Gange sind, verwenden Sie den [delete-warm-pool](#) Befehl mit der `--force-delete` Option. Diese Option beendet auch die Amazon-EC2-Instances und alle ausstehenden Lebenszyklusaktionen.

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg --force-delete
```

Auto Scaling Scaling-Gruppenzonenverschiebung

Zonal Shift ist eine Funktion im Amazon Application Recovery Controller (ARC). Mit Zonal Shift können Sie Anwendungsbeeinträchtigungen in einer Availability Zone mit einer einzigen Aktion schnell beheben. Wenn Sie Zonal Shift für eine Auto Scaling Scaling-Gruppe aktivieren, wird die Gruppe beim ARC Zonal Shift Service registriert. Anschließend können Sie mithilfe der API, oder eine Zonenverschiebung starten AWS-Managementkonsole AWS CLI, und die Auto Scaling Scaling-Gruppe behandelt die Availability Zone mit einer aktiven Zonenverschiebung als beeinträchtigt.

Auto Scaling Scaling-Konzepte für Gruppen mit zonaler Verschiebung

Bevor Sie fortfahren, stellen Sie sicher, dass Sie mit den folgenden Kernkonzepten im Zusammenhang mit der Integration mit ARC Zonal Shift vertraut sind.

ARC-Zonenverschiebung

Auto Scaling kann Auto Scaling Scaling-Gruppen mit ARC-Zonenverschiebung registrieren, wenn Sie diese Funktion aktivieren. Nach der Registrierung können Sie Ihre Ressourcen mit der [ListManagedResourcesARC-API](#) einsehen. Weitere Informationen finden Sie unter [Zonal Shift in ARC](#) im Amazon Application Recovery Controller (ARC) Developer Guide.

Neuausgleich der Availability Zone

Auto Scaling versucht, die Kapazität in jeder Availability Zone im Gleichgewicht zu halten. Wenn ein Ungleichgewicht zwischen Availability Zones auftritt, versucht Auto Scaling automatisch, das Ungleichgewicht zu beheben. Weitere Informationen finden Sie unter [Instance-Distribution](#).

Dynamische Skalierung

Die dynamische Skalierung skaliert die gewünschte Kapazität Ihrer Auto Scaling Scaling-Gruppe auf der Grundlage von Metriken, die Sie mit Skalierungsrichtlinien auswählen. Weitere Informationen finden Sie unter [Dynamische Skalierung für Amazon EC2 Auto Scaling](#).

Health checks (Zustandsprüfungen)

Auto Scaling überprüft regelmäßig den Integritätsstatus aller Instances innerhalb einer Auto Scaling Scaling-Gruppe, um sicherzustellen, dass sie laufen und in gutem Zustand sind. Wenn eine fehlerhafte Instance erkannt wird, markiert Auto Scaling sie als Ersatz. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Instance-Aktualisierung

Sie können eine Instance-Aktualisierung verwenden, um die Instances in Ihrer Auto Scaling Scaling-Gruppe zu aktualisieren. Nachdem eine Instanzaktualisierung gestartet wurde, versucht Auto Scaling, alle Instances in Ihrer Auto Scaling Scaling-Gruppe zu ersetzen. Weitere Informationen finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).

Vorskaliert

Sie können den Verlust einer einzelnen Availability Zone tolerieren, da Sie in den verbleibenden Availability Zones über genügend Kapazität für Ihre Anwendung verfügen.

Ausskalieren

Wenn Sie die gewünschte Kapazität einer Auto Scaling-Gruppe erhöhen, versucht Auto Scaling, zusätzliche Instances zu starten, um die neue gewünschte Kapazität zu erreichen. Standardmäßig startet Auto Scaling die Instance ausgewogen, um die gleiche Kapazität in jeder aktivierten Availability Zone in einer Auto Scaling Scaling-Gruppe aufrechtzuerhalten.

So funktioniert Zonal Shift für Auto Scaling Scaling-Gruppen

Angenommen, Sie haben eine Auto Scaling Scaling-Gruppe mit den folgenden Availability Zones:

- us-east-1a
- us-east-1b
- us-east-1c

Sie haben Zonal Shift in allen Availability Zones aktiviert und stellen Fehler fest, us-east-1a sodass Sie eine Zonenverschiebung auslösen. Die folgenden Verhaltensweisen treten auf, wenn eine Zonenverschiebung ausgelöst wird. us-east-1a

- Skalierung — Auto Scaling startet alle neuen Kapazitätsanfragen in den fehlerfreien Availability Zones (us-east-1b und us-east-1c).
- Dynamische Skalierung — Auto Scaling verhindert, dass Skalierungsrichtlinien die gewünschte Kapazität in allen Availability Zones verringern. Auto Scaling verhindert nicht, dass Skalierungsrichtlinien die gewünschte Kapazität in allen Availability Zones erhöhen.
- Instanzaktualisierungen — Auto Scaling verlängert das Timeout für jeden Instanzaktualisierungsprozess, der verzögert wird, während eine Zonenverschiebung aktiv ist.

In der folgenden Tabelle wird das Verhalten bei der Integritätsprüfung für jede Option beschrieben, wenn eine Zonenverschiebung ausgelöst wird. us-east-1a

Die Auswahl des Verhaltens bei der Integritätsprüfung in der Availability Zone	Verhalten bei Gesundheitschecks			
Ungesundes ersetzen	Instances, die als fehlerhaft erscheinen, werden in allen Availability Zones (us-east-1a, us-east-1b, und us-east-1c) ersetzt.			
Ungesunde Geräte ignorieren	Instanzen, die als fehlerhaft erscheinen, werden in			

Die Auswahl des Verhaltens bei der Integritätsprüfung in der Availability Zone	Verhalten bei Gesundheitschecks			
	us-east-1b und ersetzt. us-east-1c Instances in der Availability Zone werden nicht durch die aktive Zonenverschiebung () us-east-1a ersetzt.			

Bewährte Methoden für die Verwendung von Zonal Shift

Um die hohe Verfügbarkeit Ihrer Anwendungen bei Verwendung von Zonal Shift aufrechtzuerhalten, empfehlen wir die folgenden bewährten Methoden:

- Überwachen EventBridge Sie Benachrichtigungen, um festzustellen, ob eine anhaltende Beeinträchtigung der Availability Zone vorliegt. Weitere Informationen finden Sie unter [Wird EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen verwendet.](#)
- Verwenden Sie Skalierungsrichtlinien mit geeigneten Schwellenwerten, um sicherzustellen, dass Sie über genügend Kapazität verfügen, um den Verlust einer Availability Zone zu tolerieren.
- Legen Sie eine Richtlinie zur Instanzwartung fest, die mindestens einen fehlerfreien Wert von 100 vorsieht. Mit dieser Einstellung wartet Auto Scaling darauf, dass eine neue Instance einsatzbereit ist, bevor es eine fehlerhafte Instance beendet.

Für Kunden mit vorinstallierter Version empfehlen wir außerdem Folgendes:

- Wählen Sie bei der Integritätsprüfung für die beeinträchtigte Availability Zone die Option Ungesunde Instanz ignorieren aus, da Sie die fehlerhafte Instanz während des Beeinträchtigungsereignisses nicht austauschen müssen.
- Verwenden Sie Zonal Autoshift in ARC für Ihre Auto Scaling Scaling-Gruppen. Die zonale Autoshift-Funktion in ARC ermöglicht es, den Verkehr für eine Ressource von einer Availability Zone weg AWS zu verlagern, wenn eine Beeinträchtigung in einer Availability Zone AWS festgestellt wird. Weitere Informationen finden Sie unter [Zonal Autoshift in ARC](#) im Amazon Application Recovery Controller (ARC) Developer Guide.

Für Kunden mit zonenübergreifenden deaktivierten Load Balancern empfehlen wir außerdem Folgendes:

- Verwenden Sie Balanced nur für Ihre Availability Zone-Verteilung.
- Wenn Sie Zonal Shift sowohl für Auto Scaling Scaling-Gruppen als auch für Load Balancer verwenden, brechen Sie zuerst die Zonenverschiebung in Ihrer Auto Scaling Scaling-Gruppe ab. Warten Sie dann, bis die Kapazität auf alle Availability Zones verteilt ist, bevor Sie die Zonenverschiebung auf dem Load Balancer stornieren.
- Aufgrund der Möglichkeit, dass die Kapazität unausgewogen ist, wenn Sie Zonal Shift aktivieren und einen zonenübergreifenden deaktivierten Load Balancer verwenden, beinhaltet Auto Scaling einen zusätzlichen Validierungsschritt. Wenn Sie sich an bewährte Methoden halten, können Sie diese Möglichkeit bestätigen, indem Sie das AWS-Managementkonsole Kontrollkästchen aktivieren oder die `skip-zonal-shift-validation` Markierung in `CreateAutoScalingGroup`, oder verwenden. `UpdateAutoScalingGroup AttachTrafficSources`

Weitere Informationen zur Verwendung von Zonal Shift mit Auto Scaling-Gruppen finden Sie im AWS Compute-Blog [Using Zonal Shift with Amazon EC2 Auto Scaling](#).

Aktivieren Sie Zonal Shift mit dem oder AWS-Managementkonsole AWS CLI

Verwenden Sie eine der folgenden Methoden, um Zonal Shift zu aktivieren.

Console

Um Zonal Shift in einer neuen Gruppe (Konsole) zu aktivieren

1. Folgen Sie den Anweisungen unter [Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage](#) und schließen Sie jeden Schritt des Verfahrens bis zu Schritt 10 ab.
2. Aktivieren Sie auf der Seite Mit anderen Diensten integrieren für Application Recovery Controller (ARC) Zonal Shift das Kontrollkästchen, um Zonal Shift zu aktivieren.
3. Wählen Sie für Verhalten bei der Integritätsprüfung die Option Ungesund ignorieren oder Ungesund ersetzen aus. Weitere Informationen finden Sie unter [So funktioniert Zonal Shift für Auto Scaling Scaling-Gruppen](#).
4. Fahren Sie mit den Schritten unter [Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage](#) fort.

AWS CLI

Um die Zonenverschiebung für eine neue Gruppe zu aktivieren (AWS CLI)

Fügen Sie dem [create-auto-scaling-group](#)-Befehl den `--availability-zone-impairment-policy`-Parameter hinzu.

Der `--availability-zone-impairment-policy` Parameter hat zwei Optionen:

- `ZonalShiftEnabled`— Wenn auf `gesetzttrue`, registriert Auto Scaling die Auto Scaling Scaling-Gruppe mit ARC-Zonenverschiebung, [und Sie können eine Zonenverschiebung auf der ARC-Konsole starten, aktualisieren oder abbrechen](#). Wenn auf `gesetztfalse`, hebt Auto Scaling die Auto Scaling Scaling-Gruppe von ARC Zonal Shift ab. Sie müssen Zonal Shift bereits aktiviert haben, um auf setzen zu können. `false`
- `ImpairedZoneHealthCheckBehavior`— Wenn diese Option auf `gesetztistreplace-unhealthy`, werden fehlerhafte Instances in der Availability Zone durch die aktive Zonenschicht ersetzt. Wenn diese Option auf `gesetztistignore-unhealthy`, werden fehlerhafte Instances in der Availability Zone nicht durch die aktive Zonenschicht ersetzt. Weitere Informationen finden Sie unter [So funktioniert Zonal Shift für Auto Scaling Scaling-Gruppen](#).

Das folgende Beispiel aktiviert die Zonenverschiebung für eine neue Auto Scaling Scaling-Gruppe mit dem Namen `my-asg`.

```
aws autoscaling create-auto-scaling-group \  
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \  
  --auto-scaling-group-name my-asg \  
  --min-size 1 \  
  --max-size 10 \  
  --desired-capacity 5 \  
  --availability-zones us-east-1a us-east-1b us-east-1c \  
  --availability-zone-impairment-policy '{  
    "ZonalShiftEnabled": true,  
    "ImpairedZoneHealthCheckBehavior": IgnoreUnhealthy  
  }'
```

Console

Um Zonal Shift für eine bestehende Gruppe (Konsole) zu aktivieren

1. Öffnen Sie die EC2 Amazon-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die AWS-Region aus, in der Sie Ihre Auto-Scaling-Gruppe erstellt haben.
3. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Wählen Sie auf der Registerkarte Integrationen unter Application Recovery Controller (ARC) Zonal Shift die Option Bearbeiten aus.
5. Aktivieren Sie das Kontrollkästchen, um Zonal Shift zu aktivieren.
6. Wählen Sie für Verhalten bei der Integritätsprüfung die Option Ungesund ignorieren oder Ungesund ersetzen aus. Weitere Informationen finden Sie unter [So funktioniert Zonal Shift für Auto Scaling Scaling-Gruppen](#).
7. Wählen Sie Aktualisieren.

AWS CLI

Um die Zonenverschiebung für eine bestehende Gruppe zu aktivieren ()AWS CLI

Fügen Sie dem [update-auto-scaling-group](#)-Befehl den `--availability-zone-impairment-policy`-Parameter hinzu.

Der `--availability-zone-impairment-policy` Parameter hat zwei Optionen:

- `ZonalShiftEnabled`— Wenn auf `gesetzttrue`, registriert Auto Scaling die Auto Scaling Scaling-Gruppe mit ARC-Zonenverschiebung, [und Sie können eine Zonenverschiebung auf der ARC-Konsole starten, aktualisieren oder abbrechen](#). Wenn auf `gesetztfalse`, hebt Auto Scaling die Auto Scaling Scaling-Gruppe von ARC Zonal Shift ab. Sie müssen Zonal Shift bereits aktiviert haben, um auf `setzen` zu können. `false`
- `ImpairedZoneHealthCheckBehavior`— Wenn diese Option auf `gesetztistreplace-unhealthy`, werden fehlerhafte Instances in der Availability Zone durch die aktive Zonenschicht ersetzt. Wenn diese Option auf `gesetztistignore-unhealthy`, werden fehlerhafte Instances in der Availability Zone nicht durch die aktive Zonenschicht ersetzt. Weitere Informationen finden Sie unter [So funktioniert Zonal Shift für Auto Scaling Scaling-Gruppen](#).

Das folgende Beispiel aktiviert die Zonenverschiebung für die angegebene Auto Scaling Scaling-Gruppe.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--availability-zone-impairment-policy '{  
  "ZonalShiftEnabled": true,  
  "ImpairedZoneHealthCheckBehavior": IgnoreUnhealthy  
}'
```

Verteilung von Availability Zones für Auto-Scaling-Gruppen

In den folgenden Informationen werden die Availability Zone-Strategien für Auto Scaling Scaling-Gruppen beschrieben.

Ausgewogenes Bemühen

Auto Scaling sorgt für eine gleiche Anzahl von Instanzen in allen aktivierten Availability Zones. Wenn Startversuche in einer Availability Zone fehlschlagen, versucht Auto Scaling, Instances in einer anderen funktionsfähigen Availability Zone zu starten. Diese Strategie ist wichtig für Anwendungen, die Availability Zone-Redundanz benötigen und nicht durch unausgeglichene Gruppen beeinträchtigt werden.

Nur ausgewogen

Auto Scaling sorgt für eine gleiche Anzahl von Instanzen in allen aktivierten Availability Zones. Wenn Startversuche in einer Availability Zone fehlschlagen, versucht Auto Scaling weiterhin, Instances in der Availability Zone zu starten. Diese Strategie ist wichtig, um bestimmte Anforderungen zu erfüllen, z. B. quorumbasierte Workloads oder wenn Ihre Auto Scaling Scaling-Gruppe den Verlust einer Availability Zone tolerieren kann, weil Sie in den verbleibenden Availability Zones über ausreichend Kapazität verfügen.

Die Auswahl der Verteilungsstrategie für die Availability Zone befindet sich im Abschnitt Netzwerk der Befehle AWS-Managementkonsole oder Sie können die Befehle oder verwenden. [create-auto-scaling-groupupdate-auto-scaling-group](#)

Weitere Informationen finden Sie unter [Erstellen Sie Auto-Scaling-Gruppen mit Startvorlagen](#).

Instances von Ihrer Auto Scaling Scaling-Gruppe trennen oder anhängen

Sie können Instances von Ihrer Auto Scaling Scaling-Gruppe trennen. Nachdem eine Instanz getrennt wurde, wird diese Instanz unabhängig und kann entweder eigenständig verwaltet oder an eine andere Auto Scaling Scaling-Gruppe angehängt werden, unabhängig von der ursprünglichen Gruppe, zu der sie gehörte. Dies kann beispielsweise nützlich sein, wenn Sie Tests mit vorhandenen Instances durchführen möchten, auf denen Ihre Anwendung bereits ausgeführt wird.

Dieses Thema enthält Anweisungen zum Trennen und Anhängen von Instanzen. Beim Anhängen von Instanzen können Sie statt einer getrennten Instanz auch eine bestehende Instanz verwenden.

Anstatt eine Instanz zu trennen und erneut derselben Gruppe zuzuordnen, empfehlen wir, das Standby-Verfahren zu verwenden, um die Instanz vorübergehend aus der Gruppe zu entfernen. Weitere Informationen finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#).

Inhalt

- [Überlegungen zum Trennen von Instanzen](#)
- [Überlegungen zum Anhängen von Instances](#)
- [Verschieben Sie eine Instance mithilfe von Trennen und Anhängen in eine andere Gruppe](#)

Überlegungen zum Trennen von Instanzen

Beachten Sie beim Trennen von Instances die folgenden Punkte:

- Sie können eine Instanz nur trennen, wenn sie sich entweder im Status `InService` Oder `StandBy` befindet. Seien Sie vorsichtig, wenn Sie Instances trennen, die sich im `StandBy`-Zustand befinden. Wenn Sie beim Versuch, Instances zu trennen, nachdem Sie sie in den `StandBy` Status versetzt haben, das `ShouldDecrementDesiredCapacity` Flag in den API-Aufruf aufnehmen, kann dies dazu führen, dass andere Instances unerwartet beendet werden.
- Nachdem Sie eine Instance getrennt haben, läuft sie weiter und es fallen Gebühren an. Um unnötige Gebühren zu vermeiden, sollten Sie getrennte Instances erneut anhängen oder beenden, wenn sie nicht mehr benötigt werden.
- Sie können die gewünschte Kapazität um die Anzahl der Instances verringern, die Sie trennen. Wenn Sie sich dafür entscheiden, die Kapazität nicht zu verringern, startet Amazon EC2 Auto Scaling neue Instances, um die getrennten Instanzen zu ersetzen, um die gewünschte Kapazität aufrechtzuerhalten.
- Wenn die Anzahl der Instances, die Sie trennen, dazu führt, dass die Auto Scaling Scaling-Gruppe ihre Mindestkapazität unterschreitet, müssen Sie die Mindestkapazität verringern.
- Wenn Sie mehrere Instances von derselben Availability Zone trennen, ohne die gewünschte Kapazität zu verringern, nimmt die Gruppe das Gleichgewicht von selbst wieder auf, sofern Sie den Vorgang nicht unterbrechen. AZRebalance Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).
- Wird eine Instance von einer Auto-Scaling-Gruppe mit einer Load Balancer-Zielgruppe oder einem Classic Load Balancer entfernt, wird die Instance beim Load Balancer abgemeldet. Wenn Connection Draining (Abmeldeverzögerung) für Ihren Load Balancer aktiviert ist, wartet Amazon EC2 Auto Scaling auf den Abschluss von eingehenden Anforderungen.

Überlegungen zum Anhängen von Instances

Beachten Sie beim Anhängen von Instanzen Folgendes:

- Amazon EC2 Auto Scaling behandelt angehängte Instances genauso wie Instances, die von der Gruppe selbst gestartet wurden. Das bedeutet, dass angehängte Instances bei Scale-In-Ereignissen beendet werden können, sofern sie ausgewählt werden. Die von der `AWSServiceRoleForAutoScaling` serviceverknüpften Rolle gewährten Berechtigungen ermöglichen Amazon EC2 Auto Scaling, dies zu tun.

- Beim Hinzufügen von Instances erhöht sich die gewünschte Kapazität der Gruppe um die Anzahl der Instances, die hinzugefügt werden. Wenn die gewünschte Kapazität nach dem Hinzufügen der neuen Instances die maximale Gruppengröße überschreitet, schlägt die Anforderung zum Anhängen weiterer Instances fehl.
- Wenn Sie Instances zu Ihrer Gruppe hinzufügen, was zu einer ungleichmäßigen Verteilung auf die Availability Zones führt, gleicht Amazon EC2 Auto Scaling die Gruppe neu aus, um eine gleichmäßige Verteilung wiederherzustellen, sofern Sie den Vorgang nicht unterbrechen. `AZRebalance` Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).
- Wird einer Auto-Scaling-Gruppe mit einer Load Balancer-Zielgruppe oder einem Classic Load Balancer eine Instance hinzugefügt, wird die Instance beim Load Balancer registriert.

Eine zuzuweisende Instance muss die folgenden Kriterien erfüllen:

- Die Instance befindet sich bei Amazon EC2 im Zustand `running`.
- Das AMI zum Starten der Instance ist noch vorhanden.
- Die Instance gehört keiner anderen Auto-Scaling-Gruppe an.
- Die Instance wird in einer der Availability Zones gestartet, die in der Auto Scaling Scaling-Gruppe definiert sind.
- Wenn die Auto-Scaling-Gruppe über eine angefügte Load-Balancer-Zielgruppe oder einen Classic Load Balancer verfügt, müssen sich sowohl die Instance als auch der Load Balancer in derselben VPC befinden.

Verschieben Sie eine Instance mithilfe von Trennen und Anhängen in eine andere Gruppe

Verwenden Sie eines der folgenden Verfahren, um eine Instance von Ihrer Auto Scaling Scaling-Gruppe zu trennen und sie einer anderen Auto Scaling-Gruppe zuzuordnen.

Informationen zum Erstellen einer neuen Auto Scaling Scaling-Gruppe aus einer getrennten Instance finden Sie unter [Erstellen Sie eine Auto Scaling Scaling-Gruppe aus einer vorhandenen Instanz mit dem AWS CLI](#) (nicht empfohlen, erstellt eine Startkonfiguration).

Console

So trennen Sie eine Instance von einer Auto-Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Instance Management (Instance-Verwaltung) unter Instances eine Instance aus, und wählen Sie Actions (Aktionen), Detach (Trennen).
4. Lassen Sie im Dialogfeld Instanz trennen das Kontrollkästchen Instance ersetzen aktiviert, um eine Ersatz-Instance zu starten. Deaktivieren Sie dieses Kontrollkästchen, um die gewünschte Kapazität zu verringern.
5. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **detach** ein, um das Entfernen der angegebenen Instance aus der Auto-Scaling-Gruppe zu bestätigen. Wählen Sie dann Instance trennen aus.

Sie können die Instance jetzt einer anderen Auto Scaling Scaling-Gruppe zuordnen.

So fügen Sie einer Auto-Scaling-Gruppe eine Instance hinzu

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. (Optional) Wählen Sie im Navigationsbereich unter Auto Scaling die Option Auto Scaling Groups (Auto-Scaling-Gruppen) aus. Wählen Sie die Auto-Scaling-Gruppe aus und stellen Sie sicher, dass die Höchstgröße der Auto-Scaling-Gruppe ausreicht, um eine weitere Instance hinzuzufügen. Erhöhen Sie andernfalls auf der Registerkarte Details die maximale Kapazität.
3. Wählen Sie im Navigationsbereich unter Instances die Option Instances und dann eine Instance aus.
4. Wählen Sie Actions, Instance Settings und Attach to Auto Scaling Group aus.
5. Geben Sie auf der Seite Attach to Auto Scaling Group (An Auto-Scaling-Gruppe anhängen) für Auto Scaling group einen Namen für die Gruppe ein und wählen Sie anschließend Attach (Anhängen) aus.
6. Wenn die Instance die Kriterien nicht erfüllt, wird eine Fehlermeldung mit entsprechenden Details ausgegeben. Zum Beispiel befindet sich die Instance möglicherweise nicht in

derselben Availability Zone wie die Auto-Scaling-Gruppe. Wählen Sie Schließen und versuchen Sie es erneut mit einer Auto Scaling Scaling-Gruppe, die die Kriterien erfüllt.

AWS CLI

Verwenden Sie die folgenden Beispielbefehle, um eine Instanz zu trennen und anzuhängen. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

So trennen Sie eine Instance von einer Auto-Scaling-Gruppe

1. Verwenden Sie den folgenden [describe-auto-scaling-instances](#) Befehl, um die aktuellen Instanzen zu beschreiben.

```
aws autoscaling describe-auto-scaling-instances \  
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

Das folgende Beispiel zeigt die Ausgabe, die beim Ausführen dieses Befehls erzeugt wird.

Notieren Sie sich die ID der Instanz, die Sie aus der Gruppe entfernen möchten. Sie benötigen diese ID im nächsten Schritt.

```
{  
  "AutoScalingInstances": [  
    {  
      "ProtectedFromScaleIn": false,  
      "AvailabilityZone": "us-west-2a",  
      "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1",  
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
      },  
      "InstanceId": "i-05b4f7d5be44822a6",  
      "InstanceType": "t3.micro",  
      "AutoScalingGroupName": "my-asg",  
      "HealthStatus": "HEALTHY",  
      "LifecycleState": "InService"  
    },  
    {  
      "ProtectedFromScaleIn": false,  
      "AvailabilityZone": "us-west-2a",  
      "LaunchTemplate": {
```

```

        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-0c20ac468fa3049e8",
    "InstanceType": "t3.micro",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService"
},
{
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-0787762faf1c28619",
    "InstanceType": "t3.micro",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService"
},
{
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-0f280a4c58d319a8a",
    "InstanceType": "t3.micro",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService"
}
]
}

```

2. Verwenden Sie den folgenden Befehl `detach-instances`, um eine Instance zu [trennen](#), ohne die gewünschte Kapazität zu verringern.

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \  
  --auto-scaling-group-name my-asg
```

Um eine Instance zu trennen und die gewünschte Kapazität zu verringern, fügen Sie die Option hinzu. `--should-decrement-desired-capacity`

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \  
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

Sie können die Instance jetzt einer anderen Auto Scaling Scaling-Gruppe zuordnen.

So fügen Sie einer Auto-Scaling-Gruppe eine Instance hinzu

1. Verwenden Sie den folgenden Befehl [attach-instances](#), um die Instance an eine andere Auto Scaling Scaling-Gruppe anzuhängen.

```
aws autoscaling attach-instances --instance-ids i-05b4f7d5be44822a6 --auto-  
scaling-group-name my-asg-for-testing
```

2. Verwenden Sie den folgenden [describe-auto-scaling-groups](#) Befehl, um die Größe der Auto Scaling Scaling-Gruppe nach dem Anhängen einer Instance zu überprüfen.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg-  
for-testing
```

Die folgende Beispielantwort zeigt, dass die Gruppe über zwei laufende Instances verfügt, von denen eine die Instance ist, die Sie angehängt haben.

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "my-asg-for-testing",  
      "AutoScalingGroupARN": "arn",  
      "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "2",  
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
      },  
      "MinSize": 1,  
    },  
  ],  
}
```

```
    "MaxSize": 5,
    "DesiredCapacity": 2,
    ...
    "Instances": [
      {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1",
          "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-05b4f7d5be44822a6",
        "InstanceType": "t3.micro",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
      },
      {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "2",
          "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-00dcdfffd5175890",
        "InstanceType": "t3.micro",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
      }
    ],
    ...
  }
]
```

Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe

Sie können eine Instance mit dem Status `InService` in den Status `Standby` versetzen, die Instance aktualisieren oder Probleme mit ihr beheben und sie dann wieder in Betrieb nehmen. Instances im

Standby-Status sind immer noch Teil der Auto-Scaling-Gruppe, beteiligen sich aber nicht aktiv an der Verarbeitung von Load Balancer-Datenverkehr.

Diese Funktion hilft Ihnen, die Instances zu stoppen und zu starten oder sie neu zu starten, ohne sich Gedanken darüber zu machen, dass Amazon EC2 Auto Scaling die Instances im Rahmen seiner Zustandsprüfungen oder während Scale-In-Ereignissen beendet.

Sie können beispielsweise das Amazon Machine Image (AMI) einer Auto-Scaling-Gruppe jederzeit ändern, indem Sie die Startvorlage oder die Startkonfiguration ändern. Alle nachfolgenden Instances, die von der Auto-Scaling-Gruppe gestartet werden, verwenden dieses AMI. Allerdings aktualisiert die Auto-Scaling-Gruppe keine Instances, die derzeit verwendet werden. Sie können diese Instances beenden und Amazon EC2 Auto Scaling ersetzen lassen oder die Instance-Aktualisierungsfunktion verwenden, um die Instances zu beenden und zu ersetzen. Sie können auch die Instances in den Standby-Status versetzen, die Software aktualisieren und die Instances daraufhin wieder in Betrieb nehmen.

Das Trennen von Instances von einer Auto-Scaling-Gruppe ähnelt dem Setzen von Instances in den Standby-Modus. Das Trennen von Instances kann nützlich sein, wenn Sie sie einer anderen Gruppe zuordnen oder die Instances wie eigenständige EC2-Instances verwalten und sie möglicherweise beenden möchten. Weitere Informationen finden Sie unter [Instances von Ihrer Auto Scaling Scaling-Gruppe trennen oder anhängen](#).

Inhalt

- [So funktioniert der Standby-Status](#)
- [Überlegungen](#)
- [Zustand einer Instance im Standby-Status](#)
- [Entfernen Sie eine Instance vorübergehend, indem Sie sie in den Standby-Modus versetzen](#)

So funktioniert der Standby-Status

Der Standby-Status ermöglicht das vorübergehende Entfernen einer Instance aus der Auto-Scaling-Gruppe wie folgt:

1. Sie versetzen eine Instance in den Standby-Status. Die Instance verbleibt in diesem Status, bis Sie den Standby-Status beenden.
2. Ist eine Load Balancer-Zielgruppe oder ein Classic Load Balancer mit der Auto-Scaling-Gruppe verknüpft, wird die Instance vom Load Balancer abgemeldet. Ist der Connection Draining für den

Load Balancer aktiviert, wartet Elastic Load Balancing standardmäßig 300 Sekunden, bevor die Registrierung abgeschlossen wird. So können laufende Anfragen abgeschlossen werden.

3. Sie können die Instance aktualisieren oder Probleme mit ihr beheben.
4. Die Instance wird wieder in Betrieb genommen, indem der Standby-Status aufgehoben wird.
5. Ist eine Load Balancer-Zielgruppe oder ein Classic Load Balancer mit der Auto-Scaling-Gruppe verknüpft, wird die Instance mit Load Balancer angemeldet.

Weitere Informationen zum Lebenszyklus der Instances in einer Auto-Scaling-Gruppe finden Sie unter [Instance-Lebenszyklus bei Amazon EC2 Auto Scaling](#).

Überlegungen

Beim Verschieben von Instances in den Standby-Status und aus dem Standby-Status ist Folgendes zu beachten:

- Wenn Sie eine Instance in den Standby-Status versetzen, können Sie die gewünschte Kapazität entweder durch diesen Vorgang verringern oder den Wert beibehalten.
 - Wenn Sie die gewünschte Kapazität der Auto-Scaling-Gruppe nicht verringern möchten, startet Amazon EC2 Auto Scaling eine Instance, um diejenige im Standby-Status zu ersetzen. Ziel ist es, Ihnen dabei zu helfen, die Kapazität für Ihre Anwendung aufrechtzuerhalten, während sich eine oder mehrere Instances im Standby-Modus befinden.
 - Wenn Sie die gewünschte Kapazität der Auto-Scaling-Gruppe verringern möchten, wird der Start einer Instance verhindert, mit dem die Instance im Standby-Status ersetzt wird.
- Nachdem Sie die Instance wieder in Betrieb genommen haben, wird die gewünschte Kapazität entsprechend der Anzahl der Instances in der Auto-Scaling-Gruppe erhöht.
- Um das Erhöhen (und Verringern) durchführen zu können, muss die neue gewünschte Kapazität zwischen der minimalen und der maximalen Gruppengröße liegen. Andernfalls schlägt sie fehl.
- Wenn zu irgendeinem Zeitpunkt, nachdem Sie eine Instance in den Standby-Modus versetzt oder die Instance durch Verlassen des Standby-Status wieder in Betrieb genommen haben, festgestellt wird, dass Ihre Auto-Scaling-Gruppe nicht zwischen den Availability Zones ausgeglichen wird, gleicht Amazon EC2 Auto Scaling dies aus, indem es die Availability Zones neu verteilt, sofern Sie den AZRebalance-Prozess nicht unterbrechen. Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).
- Instances im Standby-Status werden Ihnen in Rechnung gestellt.

Zustand einer Instance im Standby-Status

Amazon EC2 Auto Scaling prüft den Zustand von Instances, die sich im Standby-Modus befinden, nicht. Solange sich die Instance in einem Standby-Status befindet, hat sie denselben Zustand wie vor dem Standby. Amazon EC2 Auto Scaling prüft den Zustand der Instance nicht, bis sie wieder in Betrieb genommen wird.

Wenn Sie beispielsweise eine funktionierende Instance in den Standby-Modus versetzen und dann beenden, zeigt Amazon EC2 Auto Scaling die Instance weiterhin als fehlerfrei an. Wenn Sie versuchen, die beendete Standby-Instance erneut in Betrieb zu nehmen, prüft Amazon EC2 Auto Scaling den Zustand der Instance. Falls dabei festgestellt wird, dass sie beendet wird und fehlerhaft ist, wird eine Ersatz-Instance gestartet. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Entfernen Sie eine Instance vorübergehend, indem Sie sie in den Standby-Modus versetzen

Verwenden Sie eines der folgenden Verfahren, um eine Instanz vorübergehend außer Betrieb zu setzen, indem Sie sie in den Standby-Status versetzen.

Console

So entfernen Sie eine Instance vorübergehend:

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Instance management (Instance-Verwaltung) unter Instances eine Instance aus.
4. Wählen Sie Actions, Set to Standby aus.
5. Lassen Sie im Dialogfeld Auf Standby setzen das Kontrollkästchen Instance ersetzen aktiviert, um eine Ersatz-Instance zu starten. Deaktivieren Sie dieses Kontrollkästchen, um die gewünschte Kapazität zu verringern.
6. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **standby** ein, um zu bestätigen, dass die angegebene Instance in den Status Standby versetzt wird, und wählen Sie dann Auf Standby setzen aus.

7. Sie können die Instance nach Bedarf aktualisieren oder Probleme mit ihr beheben. Wenn Sie fertig sind, fahren Sie mit dem nächsten Schritt fort, um die Instance erneut in Betrieb zu nehmen.
8. Wählen Sie die Instance aus, wählen Sie Actions, Set to InService. Wählen Sie im InService Dialogfeld „Set to“ die Option „Set to“ aus InService.

AWS CLI

Verwenden Sie die folgenden Beispielbefehle, um eine Instance vorübergehend aus Ihrer Auto Scaling Scaling-Gruppe zu entfernen. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

So entfernen Sie eine Instance vorübergehend:

1. Verwenden Sie den folgenden [describe-auto-scaling-instances](#)-Befehl, um die zu aktualisierende Instance zu identifizieren:

```
aws autoscaling describe-auto-scaling-instances \  
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

Das folgende Beispiel zeigt die Ausgabe, die beim Ausführen dieses Befehls erzeugt wird.

Notieren Sie sich die ID der Instance, die Sie aus der Gruppe entfernen möchten. Sie benötigen diese ID im nächsten Schritt.

```
{  
  "AutoScalingInstances": [  
    {  
      "ProtectedFromScaleIn": false,  
      "AvailabilityZone": "us-west-2a",  
      "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1",  
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
      },  
      "InstanceId": "i-05b4f7d5be44822a6",  
      "InstanceId": "t3.micro",  
      "AutoScalingGroupName": "my-asg",  
      "HealthStatus": "HEALTHY",  
      "LifecycleState": "InService"  
    }  
  ]  
}
```

```

    },
    ...
  ]
}

```

2. Verwenden Sie den folgenden Standbyenter-standby-Befehl, um die Instance in einen - Status zu versetzen. Die Option `--should-decrement-desired-capacity` senkt die gewünschte Kapazität, so dass die Auto-Scaling-Gruppe keine Ersatz-Instance startet.

```

aws autoscaling enter-standby --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity

```

Nachfolgend finden Sie eine Beispielantwort.

```

{
  "Activities": [
    {
      "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",
      "AutoScalingGroupName": "my-asg",
      "Description": "Moving EC2 instance to Standby:
i-05b4f7d5be44822a6",
      "Cause": "At 2023-12-15T21:31:26Z instance i-05b4f7d5be44822a6 was
moved to standby
in response to a user request, shrinking the capacity from 4 to
3.",
      "StartTime": "2023-12-15T21:31:26.150Z",
      "StatusCode": "InProgress",
      "Progress": 50,
      "Details": "{\"Subnet ID\":\"subnet-c934b782\",\"Availability Zone
\":\"us-west-2a\"}"
    }
  ]
}

```

3. (Optional) Verwenden Sie den folgenden Befehl [describe-auto-scaling-instances](#), um zu überprüfen, ob sich die Instance im Standby-Status befindet:

```

aws autoscaling describe-auto-scaling-instances --instance-
ids i-05b4f7d5be44822a6

```

Nachfolgend finden Sie eine Beispielantwort. Der Status der Instance lautet jetzt Standby.

```
{
  "AutoScalingInstances": [
    {
      "ProtectedFromScaleIn": false,
      "AvailabilityZone": "us-west-2a",
      "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
      },
      "InstanceId": "i-05b4f7d5be44822a6",
      "InstanceType": "t3.micro",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "Standby"
    },
    ...
  ]
}
```

4. Sie können die Instance nach Bedarf aktualisieren oder Probleme mit ihr beheben. Wenn Sie fertig sind, fahren Sie mit dem nächsten Schritt fort, um die Instance erneut in Betrieb zu nehmen.
5. Verwenden Sie den folgenden [exit-standby](#)-Befehl, um die Instance wieder in Betrieb zu nehmen:

```
aws autoscaling exit-standby --instance-ids i-05b4f7d5be44822a6 --auto-scaling-
group-name my-asg
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "Activities": [
    {
      "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",
      "AutoScalingGroupName": "my-asg",
      "Description": "Moving EC2 instance out of Standby:
i-05b4f7d5be44822a6",
      "Cause": "At 2023-12-15T21:46:14Z instance i-05b4f7d5be44822a6 was
moved out of standby in
```

```

        response to a user request, increasing the capacity from 3 to
4.",
        "StartTime": "2023-12-15T21:46:14.678Z",
        "StatusCode": "PreInService",
        "Progress": 30,
        "Details": "{\"Subnet ID\": \"subnet-c934b782\", \"Availability Zone
\": \"us-west-2a\"}"
    }
]
}

```

6. (Optional) Verwenden Sie den folgenden Befehl `describe-auto-scaling-instances`, um zu überprüfen, ob die Instance wieder in Betrieb ist:

```
aws autoscaling describe-auto-scaling-instances --instance-ids i-05b4f7d5be44822a6
```

Nachfolgend finden Sie eine Beispielantwort. Der Status der Instance lautet `InService`.

```

{
  "AutoScalingInstances": [
    {
      "ProtectedFromScaleIn": false,
      "AvailabilityZone": "us-west-2a",
      "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
      },
      "InstanceId": "i-05b4f7d5be44822a6",
      "InstanceType": "t3.micro",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService"
    },
    ...
  ]
}

```

Löschen der Auto-Scaling-Infrastruktur

Führen Sie die folgenden Schritte aus, um die Skalierungsinfrastruktur vollständig zu löschen.

Aufgaben

- [Löschen Ihrer Auto-Scaling-Gruppe](#)
- [\(Optional\) Löschen der Startkonfiguration](#)
- [\(Optional\) Löschen Sie die Startvorlage](#)
- [\(Optional\) Löschen des Load Balancers und der Zielgruppen](#)
- [\(Optional\) Alarme löschen CloudWatch](#)
- [Konfigurieren Sie den Löschschutz für Ihre Amazon EC2 Auto Scaling Scaling-Ressourcen](#)

Löschen Ihrer Auto-Scaling-Gruppe

Beim Löschen einer Auto-Scaling-Gruppe werden die gewünschte, minimale und maximale Größe auf 0 eingestellt. Dadurch werden die Instances beendet. Durch das Löschen einer Instance werden auch alle zugehörigen Protokolle oder Daten sowie alle Volumes der Instance gelöscht. Wenn Sie nicht möchten, dass eine oder mehrere Instances beendet werden, können Sie sie trennen, bevor Sie die Auto-Scaling-Gruppe löschen. Wenn die Gruppe Skalierungsrichtlinien besitzt, werden durch Löschen der Gruppe auch die Richtlinien, die zugrunde liegenden Alarmaktionen und alle Alarme gelöscht, denen keine Aktion mehr zugeordnet ist.

So löschen Sie die Auto-Scaling-Gruppe (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe und wählen Sie Aktionen, Löschen aus.
3. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **delete** ein, um das Löschen der angegebenen Auto-Scaling-Gruppe zu löschen, wählen Sie dann Löschen.

Ein Ladesymbol in der Spalte Name zeigt an, dass die Auto-Scaling-Gruppe gelöscht wird. Die Spalten Desired (Gewünscht), Min und Max zeigen 0-Instances der Auto-Scaling-Gruppe an. Es dauert einige Minuten, bis die Instance beendet und die Gruppe gelöscht werden. Aktualisieren Sie die Liste, um den aktuellen Status anzuzeigen.

So löschen Sie Ihre Auto-Scaling-Gruppe (AWS CLI)

Verwenden Sie den folgenden [delete-auto-scaling-group](#) Befehl, um die Auto Scaling Scaling-Gruppe zu löschen. Dieser Vorgang funktioniert nicht, wenn die Gruppe über EC2-Instances verfügt; er ist nur für Gruppen mit null Instanzen vorgesehen.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg
```

Wenn in der Gruppe Instances oder Skalierungsaktivitäten ausgeführt werden, verwenden Sie den Befehl [delete-auto-scaling-group](#) mit der Option `--force-delete`. Dadurch werden auch die EC2-Instances beendet. Wenn Sie eine Auto Scaling-Gruppe aus der Amazon EC2 Auto Scaling-Konsole löschen, verwendet die Konsole diesen Vorgang, um alle EC2-Instances zu beenden und gleichzeitig die Gruppe zu löschen.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg --force-delete
```

(Optional) Löschen der Startkonfiguration

Sie können diesen Schritt überspringen, um die Startkonfiguration für die spätere Verwendung beizubehalten.

So löschen Sie die Startkonfiguration (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im linken Navigationsbereich unter Auto Scaling Auto-Scaling-Gruppen aus.
3. Wählen Sie oben auf der Seite Startkonfigurationen aus. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Startkonfigurationen anzeigen aus, um zu bestätigen, dass Sie die Seite Startkonfigurationen aufrufen möchten.
4. Wählen Sie Ihre Startkonfiguration und anschließend Aktionen, Startkonfiguration löschen aus.
5. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Delete (Löschen).

So löschen Sie die Startkonfiguration (AWS CLI)

Verwenden Sie den folgenden [delete-launch-configuration](#)-Befehl.

```
aws autoscaling delete-launch-configuration --launch-configuration-name my-launch-config
```

(Optional) Löschen Sie die Startvorlage

Sie können Ihre Startvorlage oder nur eine Version Ihrer Startvorlage löschen. Beim Löschen einer Startvorlage werden alle Versionen davon gelöscht.

Sie können diesen Schritt überspringen, um die Startvorlage für die spätere Verwendung aufzubewahren.

So löschen Sie eine Startvorlage (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich unter Instances die Option Launch Templates aus.
3. Wählen Sie Ihre Startvorlage aus und führen Sie dann einen der folgenden Schritte aus:
 - Wählen Sie Actions (Aktionen) und Delete template (Vorlage löschen) aus. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie **Delete** ein, um das Löschen der angegebenen Auto-Scaling-Gruppe zu bestätigen, wählen Sie dann Löschen.
 - Wählen Sie Actions (Aktionen) und Delete template version (Vorlagenversion löschen) aus. Wählen Sie die zu löschende Version aus und wählen Sie Delete (Löschen).

So löschen Sie die Startvorlage (AWS CLI)

Verwenden Sie den folgenden [delete-launch-template](#)-Befehl, um Ihre Vorlage und alle Versionen davon zu löschen.

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b72934aff71
```

Alternativ können Sie den [delete-launch-template-versions](#)-Befehl verwenden, um eine bestimmte Version einer Startvorlage zu löschen.

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b72934aff71 --versions 1
```

(Optional) Löschen des Load Balancers und der Zielgruppen

Sie können diesen Schritt überspringen, wenn die Auto-Scaling-Gruppe nicht bei einem Elastic Load Balancing-Load Balancer angemeldet ist, oder wenn Sie den Load Balancer behalten und in Zukunft weiter benutzen möchten.

So löschen Sie den Load Balancer (Konsole)

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich unter LOAD BALANCING die Option Load Balancers aus.
3. Wählen Sie den Load Balancer aus und klicken Sie auf Actions (Aktionen) und dann auf Delete (Löschen).
4. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Yes, Delete (Ja, löschen).

So löschen Sie Ihre Zielgruppe (Konsole)

1. Wählen Sie im Navigationsbereich unter Load Balancing die Option Target Groups (Zielgruppen) aus.
2. Wählen Sie Ihre Zielgruppe aus und klicken Sie dann auf Actions (Aktionen), Delete (Löschen).
3. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Ja, löschen.

So löschen Sie den mit der Auto-Scaling-Gruppe verknüpften Load Balancer (AWS CLI)

Verwenden Sie für Application Load Balancers und Network Load Balancers die folgenden Befehle [delete-load-balancer](#) und [delete-target-group](#).

```
aws elbv2 delete-load-balancer --load-balancer-arn my-load-balancer-arn  
aws elbv2 delete-target-group --target-group-arn my-target-group-arn
```

Verwenden Sie für Classic Load Balancers den folgenden Befehl. [delete-load-balancer](#)

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

(Optional) Alarme löschen CloudWatch

Gehen Sie wie folgt vor, um die mit Ihrer Auto Scaling Scaling-Gruppe verknüpften CloudWatch Alarme zu löschen. So können Sie beispielsweise Alarme im Zusammenhang mit schrittweiser Skalierung oder einfachen Skalierungsrichtlinien haben.

Note

Durch das Löschen einer Auto Scaling-Gruppe werden automatisch die CloudWatch Alarme gelöscht, die Amazon EC2 Auto Scaling für eine Skalierungsrichtlinie zur Zielverfolgung verwaltet.

Sie können diesen Schritt überspringen, wenn Ihre Auto Scaling Scaling-Gruppe keinen CloudWatch Alarmen zugeordnet ist oder wenn Sie die Alarme für die future Verwendung behalten möchten.

Um die CloudWatch Alarme zu löschen (Konsole)

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Alarms (Alarme) aus.
3. Wählen Sie die Alarme aus und klicken Sie dann auf Action (Aktion), Delete (Löschen).
4. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Delete (Löschen).

Um die CloudWatch Alarme zu löschen (AWS CLI)

Verwenden Sie den [delete-alarms](#)-Befehl. Sie können einen oder mehrere Alarme gleichzeitig löschen. Sie können beispielsweise den folgenden Befehl verwenden, um die Alarme Step-Scaling-AlarmHigh-AddCapacity und Step-Scaling-AlarmLow-RemoveCapacity zu löschen.

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-RemoveCapacity
```

Konfigurieren Sie den Löschschutz für Ihre Amazon EC2 Auto Scaling Scaling-Ressourcen

Schützen Sie Ihre Amazon EC2 Auto Scaling Scaling-Infrastruktur vor versehentlichem Löschen, indem Sie mehrere Schutzebenen konfigurieren. Auto Scaling bietet mehrere Ansätze, um das ungewollte Löschen von Ressourcen für Ihre Auto Scaling Scaling-Gruppen und die von ihm verwalteten Amazon EC2 EC2-Instances zu verhindern.

Inhalt

- [Konfigurieren Sie den Schutz vor dem Löschen von Auto Scaling Scaling-Gruppen](#)

- [Steuern Sie die Löschberechtigungen mit IAM-Richtlinien](#)

Konfigurieren Sie den Schutz vor dem Löschen von Auto Scaling Scaling-Gruppen

Der Löschschutz ist eine Einstellung auf Ressourcenebene, die verhindert, dass Ihre Amazon EC2 Auto Scaling Scaling-Gruppe versehentlich gelöscht wird. Wenn diese Option aktiviert ist, verhindert der Löschschutz den Erfolg des [DeleteAutoScalingGroup](#)API-Vorgangs, sodass Sie zuerst die Löschschatzeinstellung auf eine weniger restriktive Stufe aktualisieren müssen, bevor Sie die Auto Scaling Scaling-Gruppe löschen können.

Amazon EC2 Auto Scaling bietet drei Stufen des Löschschatzes:

Keine (Standard)

Es ist kein Löschschatz aktiviert, was bedeutet, dass Ihre Auto Scaling Scaling-Gruppe mit oder ohne Verwendung der `ForceDelete` Option gelöscht werden kann. Wenn diese `ForceDelete` Option verwendet wird, werden alle Amazon EC2 EC2-Instances, die von Ihrer Auto Scaling Scaling-Gruppe verwaltet werden, ebenfalls zwangsweise beendet, ohne dass Termination-Lifecycle-Hooks ausgeführt werden.

Erzwungenes Löschen verhindern

Ihre Auto Scaling Scaling-Gruppe kann nicht gelöscht werden, wenn Sie `ForceDelete` diese Option verwenden. Diese Konfiguration ermöglicht das Löschen leerer Auto Scaling Scaling-Gruppen (Gruppen ohne Instanzen). Diese Option wird für Produktionsworkloads empfohlen, bei denen Sie eine Massenbeendigung von Instances verhindern, aber die Bereinigung leerer Gruppen zulassen möchten.

Jegliches Löschen verhindern

Ihre Auto Scaling Scaling-Gruppe kann nicht gelöscht werden, unabhängig davon, ob die `ForceDelete` Option verwendet wird. Diese Option bietet den stärksten Schutz vor versehentlichem Löschen. Sie müssen den Löschschatz explizit deaktivieren, bevor Ihre Auto Scaling Scaling-Gruppe gelöscht werden kann. Dies wird für geschäftskritische Auto Scaling Scaling-Gruppen empfohlen, die selten oder nie gelöscht werden sollten.

So funktioniert der Löschschatz

Wenn Sie versuchen, den [DeleteAutoScalingGroup](#)API-Vorgang mit aktiviertem Löschschatz durchzuführen, gehen Sie wie folgt vor:

1. Amazon EC2 Auto Scaling validiert die Löschschatzeinstellung, bevor die Anfrage verarbeitet wird.
2. Wenn die konfigurierte Löschschatzstufe den Löschversuch blockiert, gibt Amazon EC2 Auto Scaling a `ValidationError` zurück.
3. Ihre Auto Scaling Scaling-Gruppe und ihre Amazon EC2 EC2-Instances bleiben unverändert.
4. Sie müssen die Löschschatzeinstellung auf eine weniger restriktive Stufe aktualisieren, bevor Sie Ihre Auto Scaling Scaling-Gruppe löschen können.

Der Löschschatz verhindert nicht andere Operationen wie:

- Aktualisierung der Auto Scaling Scaling-Gruppenkonfiguration.
- Beenden einzelner Instanzen.
- Skalierungsvorgänge (manuell oder automatisch).
- Prozesse aussetzen oder wieder aufnehmen.

Weitere Informationen zur ordnungsgemäßen Handhabung der Instanzbeendigung finden Sie unter [Gestalten Sie Ihre Anwendungen so, dass sie die Instance-Kündigung ordnungsgemäß handhaben](#)

Konfigurieren Sie den Löschschatz

Sie können den Löschschatz einrichten, wenn Sie eine Auto Scaling Scaling-Gruppe erstellen oder die Einstellung für eine bestehende Auto Scaling Scaling-Gruppe aktualisieren.

Console

So erstellen Sie eine Auto Scaling Scaling-Gruppe mit Löschschatz

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
3. Schließen Sie die Konfigurationsschritte für Ihre Auto Scaling Scaling-Gruppe ab.
4. Erweitern Sie auf der Seite Gruppengröße und Skalierung konfigurieren die Option Zusätzliche Einstellungen.
5. Wählen Sie für den Schutz vor dem Löschen von Auto Scaling Scaling-Gruppen die gewünschte Schutzstufe aus:
 - Keiner — Kein Löschschatz (Standard)

- Erzwungenes Löschen verhindern — Blockiert erzwungenes Löschen
 - Jegliches Löschen verhindern — Alle Löschvorgänge blockieren
6. Führen Sie die verbleibenden Schritte aus, um Ihre Auto Scaling Scaling-Gruppe zu erstellen.

So aktualisieren Sie den Löschschatz für eine bestehende Auto Scaling Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.
3. Wählen Sie Aktionen und Bearbeiten.
4. Aktualisieren Sie unter Zusätzliche Einstellungen die Einstellung für den Schutz vor dem Löschen von Auto Scaling Scaling-Gruppen.
5. Wählen Sie Aktualisieren aus.

AWS CLI

So erstellen Sie eine Auto Scaling Scaling-Gruppe mit Löschschatz

Verwenden Sie den [create-auto-scaling-group](#) Befehl mit dem `--deletion-protection` Parameter:

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateName=my-template,Version='$Latest' \  
  --min-size 1 \  
  --max-size 5 \  
  --desired-capacity 2 \  
  --vpc-zone-identifier "subnet-12345678,subnet-87654321" \  
  --deletion-protection prevent-force-deletion
```

Gültige Werte für `--deletion-protection` sind: `none` | `prevent-force-deletion` | `prevent-all-deletion`

So aktualisieren Sie den Löschschatz für eine bestehende Auto Scaling Scaling-Gruppe

Verwenden Sie den [-Befehl:update-auto-scaling-group](#)

```
aws autoscaling update-auto-scaling-group \  

```

```
--auto-scaling-group-name my-asg \  
--deletion-protection prevent-all-deletion
```

Um den Löschschutz zu deaktivieren

Stellen Sie den Löschschutz ein aufnone:

```
aws autoscaling update-auto-scaling-group \  
--auto-scaling-group-name my-asg \  
--deletion-protection none
```

Um den Status des Löschschatzes zu überprüfen

Verwenden Sie den [-Befehl:describe-auto-scaling-groups](#)

```
aws autoscaling describe-auto-scaling-groups \  
--auto-scaling-group-names my-asg
```

Steuern Sie die Löschberechtigungen mit IAM-Richtlinien

Verwenden Sie AWS Identity and Access Management (IAM-) Richtlinien, um zu steuern, welche Benutzer und Rollen Auto Scaling Scaling-Gruppen löschen können. IAM-basierte Kontrollen bieten eine zusätzliche Sicherheitsebene, indem sie Berechtigungen auf Identitätsebene einschränken.

IAM-Richtlinien sind besonders nützlich, wenn Sie:

- Erlauben Sie verschiedenen Benutzern unterschiedliche Zugriffsebenen auf Auto Scaling Scaling-Operationen.
- Verhindern Sie, dass bestimmte Benutzer die ForceDelete Option verwenden, auch wenn sie andere Auto Scaling Scaling-Operationen ausführen können.
- Beschränken Sie die Löschberechtigungen auf bestimmte Auto Scaling Scaling-Gruppen.

Die folgende Richtlinie erlaubt das Löschen einer Auto-Scaling-Gruppe nur, wenn die Gruppe über das Tag verfügt **environment=development**.

JSON

```
{
```

```

"Version":"2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Action": "autoscaling:DeleteAutoScalingGroup",
  "Resource": "*",
  "Condition": {
    "StringEquals": { "aws:ResourceTag/environment": "development" }
  }
}]
}

```

Die folgende Richtlinie verwendet den `autoscaling:ForceDelete` Bedingungsschlüssel, um den Zugriff auf die `DeleteAutoScalingGroup` API-Aktion zu steuern. Dadurch kann verhindert werden, dass bestimmte Benutzer den `ForceDelete` Vorgang verwenden, der alle Amazon EC2 EC2-Instances innerhalb einer Auto Scaling Scaling-Gruppe beendet.

JSON

```

{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": "autoscaling:DeleteAutoScalingGroup",
    "Resource": "*",
    "Condition": {
      "Bool": {
        "autoscaling:ForceDelete": "true"
      }
    }
  }]
}

```

Wenn Sie keine Bedingungsschlüssel verwenden, um den Zugriff auf Auto Scaling Scaling-Gruppen zu steuern, können Sie stattdessen die Anzahl ARNs der Ressourcen in dem Resource Element angeben, um den Zugriff zu steuern.

Die folgende Richtlinie erteilt Benutzern die Erlaubnis, die `DeleteAutoScalingGroup` API-Aktion zu verwenden, jedoch nur für Auto Scaling Scaling-Gruppen, deren Name mit `beginntdevteam-`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "arn:aws:autoscaling:us-
east-1:111122223333:autoScalingGroup:*:autoScalingGroupName/devteam-*"
    }
  ]
}
```

Sie können auch mehrere angeben, ARNs indem Sie sie in eine Liste aufnehmen. Einbeziehung der UUID stellt sicher, dass Zugriff zu der spezifischen Auto Scaling-Gruppe gewährt ist. Die UUID für eine neue Gruppe unterscheidet sich von der UUID für eine gelöschte Gruppe mit demselben Namen.

```
"Resource": [
  "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-1",
  "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-2",
  "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-3"
]
```

Weitere Beispiele für IAM-Richtlinien für Amazon EC2 Auto Scaling, einschließlich Richtlinien zur Steuerung von Löschberechtigungen, finden Sie unter [Beispiele für identitätsbasierte Richtlinien](#)

Ersetzen Sie die Instances in Ihrer Auto Scaling Scaling-Gruppe

Amazon EC2 Auto Scaling bietet Funktionen, mit denen Sie die Amazon EC2 EC2-Instances in Ihrer Auto Scaling Scaling-Gruppe ersetzen können, nachdem Sie Aktualisierungen vorgenommen haben, wie z. B. das Hinzufügen einer neuen Startvorlage durch ein neues Amazon Machine Image (AMI) oder das Hinzufügen neuer Instance-Typen. Es hilft Ihnen auch dabei, Updates zu rationalisieren, indem es Ihnen die Möglichkeit gibt, sie in denselben Vorgang einzubeziehen, der die Instances ersetzt.

Dieser Abschnitt enthält Informationen, die Sie bei folgenden Aktionen unterstützen:

- Starten einer Instance-Aktualisierung, um Instances in Ihrer Auto-Scaling-Gruppe zu ersetzen.
- Deklarieren bestimmter Updates, die eine gewünschte Konfiguration beschreiben, und aktualisieren Sie die Auto-Scaling-Gruppe auf die gewünschte Konfiguration.
- Überspringen des Ersetzens bereits aktualisierter Instances.
- Verwenden Sie Checkpoints, um Instances phasenweise zu aktualisieren und Ihre Instances an bestimmten Punkten zu überprüfen.
- Verwenden Sie die Backzeit, um am Ende einer Instance-Aktualisierung eine Pause einzulegen, um den Zustand der Instance zu überprüfen.
- Erhalten von Benachrichtigungen per E-Mail, wenn ein Checkpoint erreicht ist.
- Verwenden Sie ein Rollback, um die zuvor verwendete Konfiguration der Auto-Scaling-Gruppe wiederherzustellen.
- Automatisches Rollback, wenn die Instance-Aktualisierung aus irgendeinem Grund fehlschlägt oder wenn von Ihnen angegebene CloudWatch Amazon-Alarme in den ALARM Status wechseln.
- Begrenzen Sie die Lebensdauer von Instances, um konsistente Softwareversionen und Instance-Konfigurationen in der gesamten Auto-Scaling-Gruppe bereitzustellen.
- Ersetzen Sie Root-Volumes, ohne Instances zu beenden, und behalten Sie dabei Netzwerkschnittstellen, Nicht-Root-Volumes und IAM-Richtlinien bei.

Inhalt

- [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#)

- [Auto-Scaling-Instances basierend auf der maximalen Instance-Lebensdauer ersetzen](#)

Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren

Sie können eine Instance-Aktualisierung verwenden, um die Instances in Ihrer Auto Scaling Scaling-Gruppe zu aktualisieren. Diese Funktion kann nützlich sein, wenn Sie aufgrund einer Konfigurationsänderung Instances oder deren Root-Volumes ersetzen müssen, insbesondere wenn Ihre Auto Scaling Scaling-Gruppe eine große Anzahl von Instances enthält.

Zu den Situationen, in denen eine Instanzaktualisierung hilfreich sein kann, gehören:

- Bereitstellung eines neuen Amazon Machine Image (AMI) oder Benutzerdatenskripts in Ihrer Auto Scaling Scaling-Gruppe. Sie können eine neue Startvorlage mit den Änderungen erstellen und dann eine Instance-Aktualisierung verwenden, um die Updates sofort bereitzustellen.
- Migrieren Sie Ihre Instances auf neue Instance-Typen, um von den neuesten Verbesserungen und Optimierungen zu profitieren.
- Umstellung Ihrer Auto Scaling Scaling-Gruppen von der Verwendung einer Startkonfiguration auf die Verwendung einer Startvorlage. Sie können Ihre Startkonfigurationen in Startvorlagen kopieren und dann eine Instance-Aktualisierung verwenden, um Ihre Instances auf die neuen Vorlagen zu aktualisieren. Weitere Informationen zur Migration zu Startvorlagen finden Sie unter [Migrieren Sie Ihre Auto Scaling Scaling-Gruppen, um Vorlagen zu starten](#).
- Anwenden von Sicherheitspatches oder Softwareupdates unter Beibehaltung des Instanzstatus mit langer Laufzeit und Vermeidung von Kapazitätsbeschränkungen bei speziellen Instance-Typen wie GPU- oder Mac-Instances.

Inhalt

- [So funktioniert eine Instanzaktualisierung in einer Auto Scaling Scaling-Gruppe](#)
- [Die Standardwerte für eine Instance-Aktualisierung verstehen](#)
- [Starten Sie eine Instanzaktualisierung mit dem oder AWS-Managementkonsole AWS CLI](#)
- [Überwachen Sie eine Instanzaktualisierung mit dem AWS-Managementkonsole oder AWS CLI](#)
- [Ersetzen Sie die Root-Volumes während der Instanzaktualisierung](#)
- [Brechen Sie eine Instanzaktualisierung mit dem AWS-Managementkonsole oder ab AWS CLI](#)
- [Änderungen mit manuellem oder auto Rollback rückgängig machen](#)

- [Verwenden einer Instance-Aktualisierung mit Funktion zum Überspringen des Abgleichs](#)
- [Prüfpunkte zu einer Instance-Aktualisierung hinzufügen](#)

So funktioniert eine Instanzaktualisierung in einer Auto Scaling Scaling-Gruppe

In diesem Thema wird beschrieben, wie eine Instanzaktualisierung funktioniert, und es werden die wichtigsten Konzepte vorgestellt, die Sie verstehen müssen, um sie effektiv nutzen zu können.

Inhalt

- [Funktionsweise](#)
- [Schlüsselkonzepte](#)
- [Frist der Zustandsprüfung](#)
- [Kompatibilität von Instance-Typen](#)
- [Einschränkungen](#)

Funktionsweise

Um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren, können Sie eine neue Konfiguration definieren, die die neueste Version Ihrer Anwendung und alle anderen Updates, die Sie vornehmen möchten, enthält.

Instance Refresh unterstützt zwei Strategien für die Aktualisierung von Instances:

- Rollende Strategie (Standard) — Beendet Instances und startet neue Instances stapelweise gemäß Ihren Einstellungen. Dadurch wird sichergestellt, dass Ihre Auto Scaling Scaling-Gruppe während des gesamten Aktualisierungsvorgangs die gewünschte Kapazität und Verfügbarkeit beibehält.
- Strategie zum Ersetzen des Root-Volumes — Aktualisiert Instances, indem nur das Root-Volume ersetzt wird, ohne die Instance zu beenden. Dadurch bleiben Instance-Netzwerkschnittstellen, EBS-Volumes ohne Root-Rechte und Instance-Speicherdaten erhalten.

Anforderungen an die Strategie zum Ersetzen des Root-Volumes:

- Ihre Auto Scaling Scaling-Gruppe muss eine Richtlinie für gemischte Instanzen verwenden

- Für alle Überschreibungen in der Richtlinie für gemischte Instanzen muss Folgendes angegeben werden ImageId
- AMIs darf nur ein einziges Root-Volume enthalten
- Alle Instances müssen der Konfiguration der Startvorlage der Gruppe entsprechen
- Sie müssen die Instanzaktualisierung mit einer gewünschten Konfiguration starten, die eine Richtlinie für gemischte Instanzen enthält, die ImageId Überschreibungen enthält.

Wenn Skip Matching aktiviert ist, vergleicht Auto Scaling die aktuelle AMI-ID jeder Instance mit dem AMI IDs in Ihrer gewünschten Konfiguration. Es ersetzt nur Instances, bei denen das AMI IDs nicht übereinstimmt, sodass Sie bereits aktualisierte Instances überspringen können.

Führen Sie eine Instance-Aktualisierung durch

Starten Sie eine Instanzaktualisierung, um bestehende Instanzen auf der Grundlage dieser Konfiguration durch neue zu ersetzen.

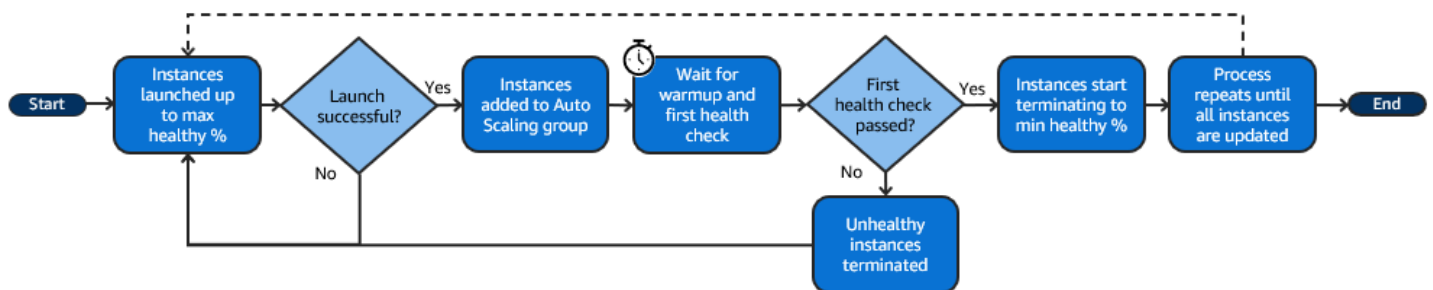
1. Erstellen Sie eine neue Startvorlage oder aktualisieren Sie die bestehende Vorlage mit den gewünschten Konfigurationsänderungen, z. B. einem neuen Amazon Machine Image (AMI). Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).
2. Starten Sie die Instance-Aktualisierung mit der Amazon EC2 Auto Scaling Scaling-Konsole oder dem SDK: AWS CLI
 - Geben Sie die neue Startvorlage oder die Version der Startvorlage an, die Sie erstellt haben. Dies wird verwendet, um neue Instances zu starten.
 - Legen Sie den bevorzugten Mindest- und Höchstwert für gesunde Werte fest. Dadurch wird gesteuert, wie viele Instances gleichzeitig ersetzt werden und ob neue Instances gestartet werden, bevor alte beendet werden.
 - Konfigurieren Sie alle optionalen Einstellungen, wie z. B.:
 - Checkpoints — Unterbrechen Sie die Aktualisierung der Instanz nach einem bestimmten Prozentsatz an Ersetzungen, um den Fortschritt zu überprüfen.
 - Backzeit — Halten Sie am Ende der Instance-Aktualisierung an, um den Zustand der Instance zu überprüfen, bevor die Instance-Aktualisierung als abgeschlossen betrachtet wird.
 - Zuordnung überspringen — Vergleicht alte Instanzen mit der neuen Konfiguration und ersetzt nur diejenigen, die nicht übereinstimmen. Wenn Sie eine Instanzaktualisierung von der Konsole aus starten, ist „Abgleich überspringen“ standardmäßig aktiviert.

- Mehrere Instanztypen — Wenden Sie eine neue oder aktualisierte [Richtlinie für gemischte Instanzen](#) als Teil der gewünschten Konfiguration an.

Wenn die Instance-Aktualisierung gestartet wurde, wird Amazon EC2 Auto Scaling:

- Ersetzt Instances stapelweise auf der Grundlage der minimalen und maximalen fehlerfreien Werte.
- Starten Sie zuerst die neuen Instances, bevor Sie die alten beenden, wenn der Mindestprozentsatz für fehlerfreie Instances auf 100 Prozent festgelegt ist. Dadurch wird sichergestellt, dass Ihre gewünschte Kapazität jederzeit beibehalten wird.
- Überprüfen Sie den Integritätsstatus der Instances und geben Sie ihnen Zeit, sich aufzuwärmen, bevor weitere Instanzen ersetzt werden.
- Beenden und ersetzen Sie Instances, die sich als fehlerhaft erwiesen haben.
- Aktualisieren Sie die Auto Scaling Scaling-Gruppeneinstellungen automatisch mit den neuen Konfigurationsänderungen, nachdem die Instanzaktualisierung erfolgreich war.
- Ersetzen Sie InService Instanzen vor Instanzen, die sich in einem warmen Pool befinden.

Das folgende Flussdiagramm veranschaulicht das Verhalten beim Starten vor dem Beenden, wenn Sie den fehlerfreien Mindestwert auf 100 Prozent festlegen.



i Note

Die Mindest- und Höchstwerte für einen fehlerfreien Zustand bei einer Instanzaktualisierung müssen nur angegeben werden, wenn Sie keine Instanzwartungsrichtlinie festgelegt haben oder wenn Sie die bestehende Richtlinie überschreiben müssen. Weitere Informationen finden Sie unter [Wartungsrichtlinien für Instances](#).

Ebenso müssen Sie den Instanz-Aufwärmzeitraum für eine Instanzaktualisierung nur angeben, wenn Sie den Standard-Warmup nicht aktiviert haben oder wenn Sie den

Standard überschreiben müssen. Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

Schlüsselkonzepte

Bevor Sie beginnen, sollten Sie sich mit folgenden zentralen Konzepten der Instance-Aktualisierungen vertraut machen:

Minimaler fehlerfreier Prozentsatz

Der minimale fehlerfreie Prozentsatz ist der Prozentsatz der gewünschten Kapazität, die während einer Instanzaktualisierung in Betrieb, fehlerfrei und einsatzbereit bleiben soll, sodass die Aktualisierung fortgesetzt werden kann. Wenn der minimale fehlerfreie Prozentsatz beispielsweise 90 Prozent beträgt, und der maximale fehlerfreie Prozentsatz 100 Prozent beträgt, dann werden jeweils 10 Prozent der Kapazität ersetzt. Wenn die neuen Instances ihre Zustandsprüfungen nicht bestehen, beendet Amazon EC2 Auto Scaling sie und ersetzt diese. Wenn die Instance-Aktualisierung keine fehlerfreien Instances starten kann, wird sie schließlich fehlschlagen, und die anderen 90 Prozent der Gruppe bleiben unberührt. Wenn die neuen Instances fehlerfrei bleiben und ihre Aufwärmphase abgeschlossen haben, kann Amazon EC2 Auto Scaling weiterhin andere Instances ersetzen.

Eine Instance-Aktualisierung kann eine einzelne Instance, mehrere Instances auf einmal oder alle auf einmal ersetzen. Um jeweils nur eine Instance zu ersetzen, legen Sie einen fehlerfreien minimalen und maximalen Prozentsatz von 100 Prozent fest. Dadurch wird das Verhalten einer Instance-Aktualisierung dahingehend geändert, dass sie vor der Beendigung gestartet wird, wodurch verhindert wird, dass die Kapazität der Gruppe unter 100 Prozent der gewünschten Kapazität fällt. Um alle Instances auf einmal zu ersetzen, legen Sie einen fehlerfreien Mindestprozentsatz von 0 Prozent fest.

Maximaler fehlerfreier Prozentsatz

Der maximale fehlerfreie Prozentsatz ist der Prozentsatz der gewünschten Kapazität, auf den Ihre Auto-Scaling-Gruppe beim Austausch von Instances erhöhen kann. Die Differenz zwischen Minimum und Maximum darf 100 nicht überschreiten. Ein größerer Bereich erhöht die Anzahl der Instances, die gleichzeitig ausgetauscht werden können.

Instance-Aufwärmphase

Der Instance-Warmup ist die Zeitspanne zwischen dem Zeitpunkt, an dem sich der Zustand einer neuen Instance zu InService ändert, und dem Zeitpunkt, an dem davon ausgegangen wird,

dass sie ihre Initialisierung abgeschlossen hat. Wenn die Instances während einer Instance-Aktualisierung ihre Zustandsprüfungen bestehen, fährt Amazon EC2 Auto Scaling nicht sofort mit dem Ersetzen der nächsten Instance fort, nachdem festgestellt wurde, dass eine neu gestartete Instance fehlerfrei ist. Es wartet die Aufwärmphase ab, bevor die nächste Instance ersetzt wird. Dies kann hilfreich sein, wenn Ihre Anwendung noch eine gewisse Initialisierungszeit benötigt, bevor sie auf Anfragen reagiert.

Die Aufwärmphase der Instance funktioniert genauso wie die standardmäßige Aufwärmphase der Instance. Daher gelten dieselben Überlegungen zur Skalierung. Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

Gewünschte Konfiguration

Die gewünschte Konfiguration ist die neue Konfiguration, die Amazon EC2 Auto Scaling in Ihrer Auto-Scaling-Gruppe bereitstellen soll. Sie können beispielsweise eine neue Startvorlage und neue Instance-Typen für Ihre Instances angeben. Während einer Instance-Aktualisierung aktualisiert Amazon EC2 Auto Scaling die Auto-Scaling-Gruppe auf die gewünschte Konfiguration. Wenn während einer Instance-Aktualisierung ein Scale-Out-Ereignis auftritt, startet Amazon EC2 Auto Scaling neue Instances mit der gewünschten Konfiguration anstelle der aktuellen Einstellungen der Gruppe. Nach erfolgreicher Instance-Aktualisierung aktualisiert Amazon EC2 Auto Scaling die Auto-Scaling-Gruppeneinstellungen, um die neue gewünschte Konfiguration wiederzugeben, die Sie als Teil der Instance-Aktualisierung angegeben haben.

Überspringen

Das Überspringen des Abgleichs weist Amazon EC2 Auto Scaling an, Instances zu ignorieren, die bereits über Ihre neuesten Aktualisierungen verfügen. Auf diese Weise ersetzen Sie nicht mehr Instances als Sie benötigen. Dies ist hilfreich, wenn Sie sicherstellen möchten, dass Ihre Auto-Scaling-Gruppe eine bestimmte Version Ihrer Startvorlage verwendet und nur die Instances ersetzt, die eine andere Version verwenden.

Prüfpunkte

Ein Checkpoint ist ein Zeitpunkt, an dem die Instance-Aktualisierung für eine bestimmte Zeit angehalten wird. Eine Instance-Aktualisierung kann mehrere Checkpoints enthalten. Amazon EC2 Auto Scaling gibt Ereignisse für jeden Checkpoint aus. Daher können Sie eine EventBridge Regel hinzufügen, um die Ereignisse an ein Ziel wie Amazon SNS zu senden, um benachrichtigt zu werden, wenn ein Checkpoint erreicht wird. Nachdem ein Prüfpunkt erreicht wurde, haben Sie die Möglichkeit, Ihre Bereitstellung zu überprüfen. Wenn Probleme festgestellt werden, können Sie die Instance -Aktualisierung abrechnen oder zurücksetzen. Die Möglichkeit, Updates in

Phasen bereitzustellen, ist ein wesentlicher Vorteil von Checkpoints. Wenn Sie keine Checkpoints verwenden, werden fortlaufend rollende Ersetzungen durchgeführt.

Weitere Informationen zu allen Standardeinstellungen, die Sie beim Starten einer Instance-Aktualisierung konfigurieren können, finden Sie unter [Die Standardwerte für eine Instance-Aktualisierung verstehen](#).

Frist der Zustandsprüfung

Amazon EC2 Auto Scaling bestimmt anhand des Status der von Ihrer Auto-Scaling-Gruppe verwendeten Zustandsprüfungen, ob eine Instance fehlerfrei ist. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Um sicherzustellen, dass diese Zustandsprüfungen so schnell wie möglich beginnen, sollten Sie die Karenzzeit für die Zustandsprüfung der Gruppe nicht zu hoch ansetzen, nur hoch genug, damit Ihre Elastic Load Balancing-Zustandsprüfungen feststellen können, ob ein Ziel zur Bearbeitung von Anfragen verfügbar ist. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).

Kompatibilität von Instance-Typen

Bevor Sie Ihren Instance-Typ ändern, sollten Sie überprüfen, ob er mit Ihrer Startvorlage kompatibel ist. Dadurch wird die Kompatibilität mit dem von Ihnen angegebenen AMI bestätigt. Angenommen, Sie haben Ihre ursprünglichen Instances von einem paravirtuellen (PV) AMI gestartet, möchten aber zu einem Instance-Typ der aktuellen Generation wechseln, der nur von einem Hardware Virtual Machine (HVM) AMI unterstützt wird. In diesem Fall müssen Sie in Ihrer Startvorlage ein HVM-AMI verwenden.

Um die Kompatibilität des Instance-Typs zu bestätigen, ohne Instances zu starten, verwenden Sie den Befehl [run-instances](#) mit der Option `--dry-run`, wie im folgenden Beispiel gezeigt.

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --dry-run
```

Informationen darüber, wie die Kompatibilität bestimmt wird, finden Sie unter [Kompatibilität bei der Änderung des Instance-Typs](#) im Amazon EC2 EC2-Benutzerhandbuch.

Einschränkungen

- Gesamtdauer — Die maximale Zeitspanne, während der eine Instance-Aktualisierung Instances weiterhin aktiv ersetzt, beträgt 14 Tage.
- Spezifisches Verhalten für gewichtete Gruppen — Wenn eine gemischte Instance-Gruppe mit einer Instance-Gewichtung konfiguriert ist, die größer oder gleich der gewünschten Kapazität der Gruppe ist, ersetzt Amazon EC2 Auto Scaling möglicherweise alle InService Instances gleichzeitig. Um diese Situation zu vermeiden, folgen Sie der Empfehlung im [Konfigurieren Sie eine Auto Scaling Scaling-Gruppe für die Verwendung von Instanzgewichten](#)-Thema. Geben Sie eine gewünschte Kapazität an, die größer ist als Ihre größte Gewichtung, wenn Sie Gewichtungen mit Ihrer Auto-Scaling-Gruppe verwenden.
- Timeout von einer Stunde — Wenn bei einer Instance-Aktualisierung keine weiteren Ersetzungen vorgenommen werden können, weil sie darauf wartet, Instances im Standby-Modus oder vor dem Skalieren geschützt zu ersetzen, oder wenn die neuen Instances ihre Zustandsprüfungen nicht bestehen, versucht Amazon EC2 Auto Scaling es eine Stunde lang erneut. Es wird auch eine Statusmeldung angezeigt, mit der Sie das Problem beheben können. Wenn das Problem nach einer Stunde weiterhin besteht, schlägt der Vorgang fehl. Die Absicht besteht darin, ihm im Falle eines vorübergehenden Problems Zeit zur Wiederherstellung zu geben.
- Code mithilfe von Benutzerdaten bereitstellen — Beim Skip Matching wird nicht nach Codeänderungen gesucht, die über ein Benutzerdatenskript bereitgestellt werden. Wenn Sie Benutzerdaten verwenden, um neuen Code abzurufen und diese Updates auf neuen Instances zu installieren, empfehlen wir Ihnen, den Skip-Abgleich zu deaktivieren, um sicherzustellen, dass alle Instanzen Ihren neuesten Code erhalten, auch ohne ein Versionsupdate für die Startvorlage.
- Aktualisierungseinschränkung — Wenn Sie versuchen, die Startvorlage, die Startkonfiguration oder die Richtlinie für gemischte Instanzen einer Auto Scaling Scaling-Gruppe zu aktualisieren, während eine Instance-Aktualisierung mit der gewünschten Konfiguration aktiv ist, schlägt die Anfrage mit dem folgenden Validierungsfehler fehl: `An active instance refresh with a desired configuration exists. All configuration options derived from the desired configuration are not available for update while the instance refresh is active.`
- Attributbasierte Instanzauswahl — Wenn Ihre Auto Scaling Scaling-Gruppe die attributbasierte Instanzauswahl verwendet (InstanceRequirements in einer Richtlinie für gemischte Instanzen spezifiziert), werden die folgenden Instanzaktualisierungsparameter nicht unterstützt:

- **SkipMatching**- Eine Instanzaktualisierung schlägt fehl, wenn Sie sie mit dem **SkipMatching** Parameter in einer **InstanceRequirements** konfigurierten Auto Scaling Scaling-Gruppe starten.
- **DesiredConfiguration**- Eine Instanzaktualisierung schlägt fehl, wenn Sie sie mit dem **DesiredConfiguration** Parameter in einer **InstanceRequirements** konfigurierten Auto Scaling Scaling-Gruppe starten.

Wenn Sie eine Instanzaktualisierung für eine Auto Scaling Scaling-Gruppe mit attributbasierter Instanzauswahl durchführen müssen, starten Sie die Instanzaktualisierung ohne diese Parameter.

- Der Austausch von Root-Volumes unterstützt Elastic Load Balancing nicht.

Die Standardwerte für eine Instance-Aktualisierung verstehen

Bevor Sie eine Instance-Aktualisierung starten, können Sie verschiedene Voreinstellungen anpassen, die sich auf die Instance-Aktualisierung auswirken. Einige Voreinstellungen sind unterschiedlich, je nachdem, ob Sie die Konsole oder die Befehlszeile (AWS CLI oder das AWS SDK) verwenden.

In der folgenden Tabelle sind die Standardwerte für Einstellungen der Instance-Aktualisierung aufgeführt.

Einstellung	AWS CLI oder SDK AWS	Die Konsole von Amazon EC2 Auto Scaling
CloudWatch Alarm	Deaktiviert (null)	Disabled
Automatisches Zurücksetzen	Deaktiviert (false)	Disabled
Zeit zum Backen	Null	Null
Prüfpunkte	Deaktiviert (false)	Disabled
Checkpoint-Verzögerung	1 Stunde (3600 Sekunden)	1 Stunde
Instance-Aufwärmphase	Die standardmäßige Instance-Aufwärmphase , falls definiert , oder andernfalls die Frist für die Zustandsprüfung .	Die standardmäßige Instance-Aufwärmphase , falls definiert , oder andernfalls die Frist für die Zustandsprüfung .

Einstellung	AWS CLI oder SDK AWS	Die Konsole von Amazon EC2 Auto Scaling
Maximaler fehlerfreier Prozentsatz	Variiert je nach Ihrer Instance-Wartungsrichtlinie. Wenn es keine Instanz-Wartungsrichtlinie gibt, ist diese standardmäßig auf 100 Prozent (Null) eingestellt.	Variiert je nach Ihrer Instance-Wartungsrichtlinie. Wenn es keine Instanz-Wartungsrichtlinie gibt, ist diese standardmäßig auf 100 Prozent (Null) eingestellt.
Minimaler fehlerfreier Prozentsatz	Variiert je nach Ihrer Instance-Wartungsrichtlinie. Wenn keine Instance-Wartungsrichtlinie vorhanden ist, wird dieser Wert standardmäßig auf 90 Prozent gesetzt.	Variiert je nach Ihrer Instance-Wartungsrichtlinie. Wenn keine Instance-Wartungsrichtlinie vorhanden ist, wird dieser Wert standardmäßig auf 90 Prozent gesetzt.
Vor Abskalierung geschützte Instances	Wait	Ignore
Überspringen	Deaktiviert (<code>false</code>)	Aktiviert
Standby-Instances	Wait	Ignore

Es folgt eine Beschreibung der einzelnen Einstellungen:

CloudWatch Alarm (**AlarmSpecification**)

Die CloudWatch Alarmspezifikation. CloudWatch Alarme können verwendet werden, um Probleme zu identifizieren und den Vorgang fehlschlagen zu lassen, wenn ein Alarm in den ALARM Status wechselt. Weitere Informationen finden Sie unter [Starten einer Instance-Aktualisierung mit automatischem Rollback](#).

Automatisches Zurücksetzen (**AutoRollback**)

Legt fest, ob Amazon EC2 Auto Scaling die Auto-Scaling-Gruppe auf ihre vorherige Konfiguration zurücksetzt, wenn die Aktualisierung der Instance fehlschlägt. Weitere Informationen finden Sie unter [Änderungen mit manuellem oder auto Rollback rückgängig machen](#).

Backzeit (**BakeTime**)

Die Wartezeit am Ende einer Instanzaktualisierung, bevor die Instanzaktualisierung als abgeschlossen betrachtet wird.

Checkpoints (**CheckpointPercentages**)

Steuert, ob Amazon EC2 Auto Scaling Instances phasenweise ersetzt. Dies ist nützlich, wenn Sie Ihre Instances überprüfen müssen, bevor Sie alle Instances austauschen. Weitere Informationen finden Sie unter [Prüfpunkte zu einer Instance-Aktualisierung hinzufügen](#).

Checkpoint-Verzögerung (**CheckpointDelay**)

Die Zeit in Sekunden, die nach einem Checkpoint gewartet werden muss, bevor fortgefahren wird. Weitere Informationen finden Sie unter [Prüfpunkte zu einer Instance-Aktualisierung hinzufügen](#).

Instance-Aufwärmphase (**InstanceWarmup**)

Eine Zeitspanne in Sekunden, in der Amazon EC2 Auto Scaling wartet, bis die Initialisierung einer neuen Instance als abgeschlossen gilt, bevor die nächste Instance ersetzt wird. Wenn Sie bereits eine standardmäßige Aufwärmphase für die Instance der Auto-Scaling-Gruppe definiert haben, müssen Sie die Aufwärmphase für die Instance nicht ändern (es sei denn, Sie möchten den Standard überschreiben). Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

Maximaler fehlerfreier Prozentsatz (**MaxHealthyPercentage**)

Der Prozentsatz der gewünschten Kapazität der Auto-Scaling-Gruppe, auf den sich Ihre Gruppe beim Ersetzen von Instances erhöhen kann.

Minimaler fehlerfreier Prozentsatz (**MinHealthyPercentage**)

Der Prozentsatz der gewünschten Kapazität der Auto-Scaling-Gruppe, der betriebsbereit, fehlerfrei und einsatzbereit sein muss, bevor der Vorgang fortgesetzt werden kann.

Vor Abskalierung geschützte Instances (**ScaleInProtectedInstances**)

Steuert, was Amazon EC2 Auto Scaling macht, wenn Instances gefunden werden, die vor dem Abskalieren geschützt sind. Weitere Informationen zu diesen Instances finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).

Amazon EC2 Auto Scaling bietet die folgenden Optionen an:

- **Replace (Refresh)** — Ersetzt Instances, die vor einer Skalierung geschützt sind.

- Ignorieren (Ignore) — Ignoriert Instances, die vor einer Skalierung geschützt sind, und ersetzt weiterhin Instances, die nicht geschützt sind.
- Warten (Wait) — Wartet eine Stunde, bis Sie den Scale-In-Schutz entfernt haben. Wenn Sie dies nicht tun, schlägt die Instance-Aktualisierung fehl.

Überspringen des Abgleichs (**SkipMatching**)

Steuert, ob Amazon EC2 Auto Scaling das Ersetzen von Instances, die der gewünschten Konfiguration entsprechen, überspringt. Wenn keine gewünschte Konfiguration angegeben wird, werden Instances mit der gleichen Startvorlage und den gleichen Instance-Typen, die die Auto-Scaling-Gruppe vor dem Start der Instance-Aktualisierung verwendet hat, nicht ersetzt. Weitere Informationen finden Sie unter [Verwenden einer Instance-Aktualisierung mit Funktion zum Überspringen des Abgleichs](#).

Standby-Instances (**StandbyInstances**)

Steuert, was Amazon EC2 Auto Scaling macht, wenn sich Instances im Zustand Standby befinden. Weitere Informationen zu diesen Instances finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#).

Amazon EC2 Auto Scaling bietet die folgenden Optionen an:

- Terminate (Terminate) — Beendet Instances, die aktiv sind. Standby
- Ignorieren (Ignore) — Ignoriert Instanzen, die sich im Status befinden, Standby und ersetzt weiterhin Instanzen, die sich im Status befinden. InService
- Warten (Wait) — Wartet eine Stunde, bis Sie die Instances wieder in Betrieb nehmen. Wenn Sie dies nicht tun, schlägt die Instance-Aktualisierung fehl.

Starten Sie eine Instanzaktualisierung mit dem oder AWS-Managementkonsole AWS CLI

Important

Sie können eine Instance-Aktualisierung, die gerade ausgeführt wird, zurücksetzen, um alle Änderungen rückgängig zu machen. Damit dies funktioniert, muss die Auto-Scaling-Gruppe die Voraussetzungen für die Verwendung von Rollbacks erfüllen, bevor die Instance-Aktualisierung gestartet wird. Weitere Informationen finden Sie unter [Änderungen mit manuellem oder auto Rollback rückgängig machen](#).

Die folgenden Verfahren helfen Ihnen dabei, eine Instanzaktualisierung mithilfe von AWS-Managementkonsole oder zu starten AWS CLI.

Starten einer Instance-Aktualisierung (Konsole)

Durch das erste Starten einer Instance-Aktualisierung mit der Konsole werden Sie die verfügbaren Funktionen und Optionen besser verstehen.

Starten einer Instance-Aktualisierung in der Konsole (grundlegendes Verfahren)

Gehen Sie wie folgt vor, wenn Sie zuvor keine [mixed instances policy](#) (Richtlinie für gemischte Instances) für Ihre Auto-Scaling-Gruppe definiert haben. Wenn Sie zuvor eine Richtlinie für gemischte Instances definiert haben, lesen Sie [Starten einer Instance-Aktualisierung in der Konsole \(Gruppe mit gemischten Instances\)](#) zum Starten einer Instance-Aktualisierung.

So starten Sie eine Instance-Aktualisierung

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Unten auf der Seite Auto-Scaling-Gruppen wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Instance refresh (Instance-Aktualisierung) unter Active instance refresh (Aktive Instance-Aktualisierung) die Option Start instance refresh (Instance-Aktualisierung starten) aus.
4. Für die Verfügbarkeitseinstellungen gehen Sie wie folgt vor:
 - a. Für die Methode der Instance-Ersetzung:
 - Wenn Sie keine Instance-Wartungsrichtlinie für die Auto-Scaling-Gruppe festgelegt haben, lautet die Standardeinstellung für die Methode zum Ersetzen von Instances Beenden und starten. Dies ist das alte Standardverhalten für eine Instance-Aktualisierung.
 - Wenn Sie eine Instance-Wartungsrichtlinie für die Auto-Scaling-Gruppe festlegen, bietet diese Standardwerte für die Instance-Ersatzmethode. Um die Instance-Wartungsrichtlinie zu überschreiben, wählen Sie Override. Überschreiben gilt nur für die aktuelle Instance-Aktualisierung. Wenn Sie das nächste Mal eine Instance-Aktualisierung starten, werden diese Werte auf die Standardwerte der Instance-Wartungsrichtlinie zurückgesetzt.

Im folgenden Verfahren wird erläutert, wie die Instance-Ersatzmethode aktualisiert werden kann.

- i. Wählen Sie eine der folgenden Instance-Ersatzmethoden:
 - Vor dem Beenden starten: Eine neue Instance muss zuerst bereitgestellt werden, bevor eine bestehende Instance beendet werden kann. Dies ist eine gute Wahl für Anwendungen, bei denen Verfügbarkeit wichtiger ist als Kosteneinsparungen.
 - Beenden und starten: Neue Instances werden zur gleichen Zeit bereitgestellt, wie Ihre bestehenden Instances beendet werden. Dies ist eine gute Wahl für Anwendungen, bei denen Kosteneinsparungen Vorrang vor der Verfügbarkeit haben. Es ist auch eine gute Wahl für Anwendungen, die nicht mehr Kapazität benötigen, als derzeit verfügbar ist.
 - Benutzerdefiniertes Verhalten: Mit dieser Option können Sie einen benutzerdefinierten Mindest- und Höchstbereich für die Kapazität einrichten, die beim Austausch von Instances verfügbar sein soll. Dies kann Ihnen helfen, das richtige Gleichgewicht zwischen Kosten und Verfügbarkeit zu finden.
- ii. Geben Sie unter Fehlerfreien Prozentsatz festlegen Werte für eines oder beide der folgenden Felder ein. Die Aktivierungsfelder variieren je nach gewählter Option für die Instance-Ersatzmethode.
 - Min.: Legt den fehlerfreien Mindestprozentsatz fest, der erforderlich ist, um mit der Instance-Aktualisierung fortzufahren.
 - Max.: Legt den maximalen fehlerfreien Prozentsatz fest, der während der Instance-Aktualisierung möglich ist.
- iii. Erweitern Sie den Abschnitt Geschätzte temporäre Kapazität bei Austauscharbeiten auf der Grundlage der aktuellen Gruppengröße anzuzeigen, um zu überprüfen, ob die Werte für Min. und Max für Ihre Gruppe gelten. Welche genauen Werte verwendet werden, hängt vom gewünschten Kapazitätswert ab, der sich ändert, wenn die Gruppe skaliert wird.
- iv. Erweitern Sie den Abschnitt Fallback-Verhalten für ungültige Ersatzgrößen festlegen und wählen Sie dann aus, ob Sie gegen den maximalen fehlerfreien Prozentsatz verstoßen möchten, um der Verfügbarkeit Priorität einzuräumen, oder ob Sie gegen den minimalen fehlerfreien Prozentsatz verstoßen möchten.

Es wird für sehr kleine Gruppen nicht empfohlen, die Standardoption Minimaler fehlerfreie Prozentsatz beizubehalten. Wenn sich nur eine Instance in der Auto-Scaling-Gruppe befindet, kann das Starten einer Instance-Aktualisierung zu einem Ausfall führen.

Dieser Schritt konfiguriert das Fallback-Verhalten, wenn Sie eine Auto-Scaling-Gruppe verwenden, die noch keine Instance-Wartungsrichtlinie hat. Diese Option ist nicht verfügbar und wird nicht angezeigt, wenn Ihre Gruppe über eine Instance-Wartungsrichtlinie verfügt. Diese Option ist auch nur für die Ersatzmethode Beenden und Starten verfügbar. Bei anderen Ersatzmethoden wird die maximale Fehlerquote überschritten, um der Verfügbarkeit Priorität einzuräumen.

- b. Geben Sie für Instance-Aufwärmphase die Anzahl der Sekunden ein, ab der sich der Status einer neuen Instance in `InService` ändert, wenn sie ihre Initialisierung abgeschlossen hat. Amazon EC2 Auto Scaling wartet diese Zeit ab, bevor es mit dem Ersetzen der nächsten Instance fortfährt.

Während der Aufwärmphase wird eine neu gestartete Instance nicht zu den aggregierten Metriken der Auto-Scaling-Gruppe (z. B. `CPUUtilization`, `NetworkIn` und `NetworkOut`) gezählt. Wenn Sie der Auto-Scaling-Gruppe Skalierungsrichtlinien hinzugefügt haben, werden die Skalierungsaktivitäten parallel ausgeführt. Wenn Sie ein langes Intervall für die Aufwärmphase der Instance-Aktualisierung festlegen, dauert es länger, bis neu gestartete Instances in den Metriken angezeigt werden. Daher verhindert eine angemessene Aufwärmphase, dass Amazon EC2 Auto Scaling auf veraltete Metrikdaten skaliert.


Wenn Sie bereits eine standardmäßige Aufwärmphase für die Instance der Auto-Scaling-Gruppe definiert haben, müssen Sie die Aufwärmphase für die Instance nicht ändern. Wenn Sie die Standardeinstellung jedoch überschreiben möchten, können Sie einen Wert für diese Option festlegen. Weitere Informationen zum Festlegen der standardmäßigen Aufwärmphase der Instance finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

5. Für Einstellungen aktualisieren nehmen Sie Folgendes vor:
 - a. (Optional) Für Checkpoints (Prüfpunkte) wählen Sie `Enable Checkpoints` (Prüfpunkte aktivieren) aus, um Instances mit einem inkrementellen oder schrittweisen Ansatz für eine Instance-Aktualisierung zu ersetzen. Dies bietet zusätzliche Zeit für die Verifizierung verschiedener Ersatzvorgänge. Wenn Sie sich dafür entscheiden, Prüfpunkte nicht zu aktivieren, werden die Instances in einem fast kontinuierlichen Vorgang ersetzt.

Wenn Sie Prüfpunkte aktivieren, finden Sie unter [Aktivieren von Prüfpunkten \(Konsole\)](#) zusätzliche Schritte.

- b. (Optional) Geben Sie unter Backzeit an, wie lange am Ende der Instance-Aktualisierung gewartet werden soll, bevor die Instance-Aktualisierung als abgeschlossen betrachtet wird.
- c. Aktivieren oder Deaktivieren von Skip Matching (Abgleich überspringen):
 - Um das Ersetzen von Instances zu überspringen, die bereits mit Ihrer Startvorlage übereinstimmen, lassen Sie das Kontrollkästchen Überspringen des Abgleichs aktivieren aktiviert.
 - Wenn Sie das Überspringen des Abgleichs deaktivieren, indem Sie dieses Kontrollkästchen deaktivieren, können alle Instances ersetzt werden.

Wenn Sie das Überspringen des Abgleichs aktivieren, können Sie eine neue Startvorlage oder eine neue Version der Startvorlage festlegen, anstatt die vorhandene zu verwenden. Sie können dies im Abschnitt Gewünschte Konfiguration auf der Seite Instance-Aktualisierung starten tun.

 Note

Um die Funktion „Abgleich überspringen“ zur Aktualisierung einer Auto-Scaling-Gruppe zu verwenden, die derzeit eine Startkonfiguration verwendet, müssen Sie eine Startvorlage in Desired configuration (Gewünschte Konfiguration) auswählen. Die Verwendung von „Abgleich überspringen“ mit einer Startkonfiguration wird nicht unterstützt.

- d. Wählen Sie für Standby-Instances die Option Ignorieren, Beenden oder Warten aus. Dies legt fest, was passiert, wenn Instances im Standby-Status erkannt werden. Weitere Informationen finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#).

Wenn Sie Warten auswählen, müssen Sie zusätzliche Schritte unternehmen, um diese Instances wieder in Betrieb zu nehmen. Wenn Sie dies nicht tun, ersetzt die Instance-Aktualisierung alle InService-Instances und wartet eine Stunde. Wenn dann noch Standby-Instances übrig bleiben, schlägt die Instance-Aktualisierung fehl. Um diese Situation zu vermeiden, wählen Sie für diese Instances stattdessen Ignorieren oder Beenden aus.

- e. Wählen Sie für Vor Abskalierung geschützte Instances die Option Ignorieren, Ersetzen oder Warten aus. Dies legt fest, was passiert, wenn vor Abskalierung geschützte Instances erkannt werden. Weitere Informationen finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).

Wenn Sie Warten auswählen, müssen Sie zusätzliche Schritte unternehmen, um den Abskalierungsschutz dieser Instances zu entfernen. Wenn Sie dies nicht tun, ersetzt die Instance-Aktualisierung alle ungeschützten Instances und wartet eine Stunde. Wenn dann vor Abskalierung geschützte Instances übrig bleiben, schlägt die Instance-Aktualisierung fehl. Um diese Situation zu verhindern, wählen Sie stattdessen für diese Instances Ignorieren oder Ersetzen aus.

6. (Optional) Wählen Sie für CloudWatch Alarm die Option CloudWatch Alarme aktivieren und wählen Sie dann einen oder mehrere Alarme aus. CloudWatch Alarme können verwendet werden, um Probleme zu identifizieren und den Vorgang fehlschlagen zu lassen, wenn ein Alarm in den ALARM Status wechselt. Weitere Informationen finden Sie unter [Starten einer Instance-Aktualisierung mit automatischem Rollback](#).
7. (Optional) Erweitern Sie den Abschnitt Gewünschte Konfiguration, um Aktualisierungen anzugeben, die Sie an Ihrer Auto-Scaling-Gruppe vornehmen möchten.


Für diesen Schritt können Sie die JSON- oder YAML-Syntax verwenden, um Parameterwerte zu bearbeiten, anstatt eine Auswahl in der Konsolenschnittstelle zu treffen. Wählen Sie hierfür Use code editor (Code-Editor verwenden) anstelle von Use console interface (Konsolenschnittstelle verwenden) aus. Im folgenden Verfahren wird erläutert, wie mit der Konsolenschnittstelle eine Auswahl getroffen werden kann.

- a. Für update launch template (Startvorlage aktualisieren):
 - Wenn Sie keine neue Startvorlage oder neue Startvorlagenversion für Ihre Auto-Scaling-Gruppe erstellt haben, aktivieren Sie dieses Kontrollkästchen nicht.
 - Wenn Sie eine neue Startvorlage oder eine neue Startvorlagen-Version erstellt haben, aktivieren Sie dieses Kontrollkästchen. Wenn Sie diese Option auswählen, zeigt Amazon EC2 Auto Scaling Ihnen die aktuelle Startvorlage und die aktuelle Version der Startvorlage an. Es listet auch alle anderen verfügbaren Versionen auf. Wählen Sie die Startvorlage und dann die Version aus.

Nachdem Sie eine Version ausgewählt haben, werden Ihnen die Versionsinformationen angezeigt. Dies ist die Version der Startvorlage, die beim Ersetzen von Instances im


Rahmen einer Instance-Aktualisierung verwendet wird. Wenn die Instance-Aktualisierung erfolgreich ist, wird diese Version der Startvorlage auch verwendet, wenn neue Instances gestartet werden, z. B. wenn die Gruppe skaliert wird.

- b. Für Choose a set of instance types and purchase options to override the instance type in the launch template (Eine Reihe von Instance-Typen und Kaufoptionen auswählen, um den Instance-Typ in der Startvorlage außer Kraft zu setzen):
- Aktivieren Sie dieses Kontrollkästchen nicht, wenn Sie den Instance-Typ und die Kaufoption verwenden möchten, die Sie in Ihrer Startvorlage angegeben haben.
 - Aktivieren Sie dieses Kontrollkästchen, wenn Sie den Instance-Typ in der Startvorlage überschreiben oder Spot-Instances ausführen möchten. Sie können jeden Instance-Typ entweder manuell hinzufügen oder einen primären Instance-Typ und eine Empfehlungsoption auswählen, die alle zusätzlichen passenden Instance-Typen für Sie abrufen. Wenn Sie Spot-Instances starten möchten, empfehlen wir, einige unterschiedliche Instance-Typen hinzuzufügen. Dadurch kann Amazon EC2 Auto Scaling einen weiteren Instance-Typ starten, wenn in den ausgewählten Availability Zones nicht genügend Instance-Kapazität zur Verfügung steht. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

 Warning

Verwenden Sie Spot-Instances nicht mit Anwendungen, die eine Spot-Instance-Unterbrechung nicht verarbeiten können. Unterbrechungen können auftreten, wenn der Amazon-EC2-Spot-Service Kapazität zurückfordern muss.

Wenn Sie dieses Kontrollkästchen auswählen, stellen Sie sicher, dass die Startvorlage nicht bereits Spot-Instances anfordert. Sie können keine Startvorlage verwenden, die Spot-Instances zum Erstellen einer Auto-Scaling-Gruppe auffordert, die mehrere Instance-Typen verwendet und Spot- und On-Demand-Instances startet.

 Note

Um diese Optionen in einer Auto-Scaling-Gruppe zu aktualisieren, die derzeit eine Startkonfiguration verwendet, müssen Sie eine Startvorlage in Update launch

Vorlage (Startvorlage aktualisieren) auswählen. Das Überschreiben des Instance-Typs in Ihrer Startkonfiguration wird nicht unterstützt.

8. (Optional) Wählen Sie unter Rollback-Einstellungen die Option Automatisches Rollback aktivieren aus, um die Instance-Aktualisierung automatisch zurückzusetzen, falls sie fehlschlägt.

Diese Einstellung kann nur aktiviert werden, wenn die Auto-Scaling-Gruppe die Voraussetzungen für die Verwendung von Rollbacks erfüllt.

Weitere Informationen finden Sie unter [Änderungen mit manuellem oder auto Rollback rückgängig machen](#).
9. Überprüfen Sie Ihre gesamte Auswahl, um zu bestätigen, dass alles korrekt eingerichtet ist.

An dieser Stelle empfiehlt es sich sicherzustellen, dass sich die Unterschiede zwischen den aktuellen und den vorgeschlagenen Änderungen nicht auf unerwartete oder unerwünschte Weise auf Ihre Anwendung auswirken. Informationen zur Bestätigung, dass Ihr Instance-Typ mit Ihrer Startvorlage kompatibel ist, finden Sie unter [Kompatibilität von Instance-Typen](#).
10. Wenn Sie mit der Auswahl für Ihre Instance-Aktualisierung zufrieden sind, klicken Sie auf Instance-Aktualisierung Starten.

Starten einer Instance-Aktualisierung in der Konsole (Gruppe mit gemischten Instances)

Gehen Sie wie folgt vor, wenn Sie eine Auto-Scaling-Gruppe mit einer [Richtlinie für gemischte Instances](#) erstellt haben. Wenn Sie zuvor keine Richtlinie für gemischte Instances für Ihre Gruppe definiert haben, lesen Sie [Starten einer Instance-Aktualisierung in der Konsole \(grundlegendes Verfahren\)](#) zum Starten einer Instance-Aktualisierung.

So starten Sie eine Instance-Aktualisierung

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Unten auf der Seite Auto-Scaling-Gruppen wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Instance refresh (Instance-Aktualisierung) unter Active instance refresh (Aktive Instance-Aktualisierung) die Option Start instance refresh (Instance-Aktualisierung starten) aus.
4. Für die Verfügbarkeitseinstellungen gehen Sie wie folgt vor:

a. Für die Methode der Instance-Ersetzung:

- Wenn Sie keine Instance-Wartungsrichtlinie für die Auto-Scaling-Gruppe festgelegt haben, lautet die Standardeinstellung für die Methode zum Ersetzen von Instances Beenden und starten. Dies ist das alte Standardverhalten für eine Instance-Aktualisierung.
- Wenn Sie eine Instance-Wartungsrichtlinie für die Auto-Scaling-Gruppe festlegen, bietet diese Standardwerte für die Instance-Ersatzmethode. Um die Instance-Wartungsrichtlinie zu überschreiben, wählen Sie Override. Überschreiben gilt nur für die aktuelle Instance-Aktualisierung. Wenn Sie das nächste Mal eine Instance-Aktualisierung starten, werden diese Werte auf die Standardwerte der Instance-Wartungsrichtlinie zurückgesetzt.

Im folgenden Verfahren wird erläutert, wie die Instance-Ersatzmethode aktualisiert werden kann.

i. Wählen Sie eine der folgenden Instance-Ersatzmethoden:

- Vor dem Beenden starten: Eine neue Instance muss zuerst bereitgestellt werden, bevor eine bestehende Instance beendet werden kann. Dies ist eine gute Wahl für Anwendungen, bei denen Verfügbarkeit wichtiger ist als Kosteneinsparungen.
- Beenden und starten: Neue Instances werden zur gleichen Zeit bereitgestellt, wie Ihre bestehenden Instances beendet werden. Dies ist eine gute Wahl für Anwendungen, bei denen Kosteneinsparungen Vorrang vor der Verfügbarkeit haben. Es ist auch eine gute Wahl für Anwendungen, die nicht mehr Kapazität benötigen, als derzeit verfügbar ist.
- Benutzerdefiniertes Verhalten: Mit dieser Option können Sie einen benutzerdefinierten Mindest- und Höchstbereich für die Kapazität einrichten, die beim Austausch von Instances verfügbar sein soll. Dies kann Ihnen helfen, das richtige Gleichgewicht zwischen Kosten und Verfügbarkeit zu finden.

ii. Geben Sie unter Fehlerfreien Prozentsatz festlegen Werte für eines oder beide der folgenden Felder ein. Die Aktivierungsfelder variieren je nach gewählter Option für die Instance-Ersatzmethode.

- Min.: Legt den fehlerfreien Mindestprozentsatz fest, der erforderlich ist, um mit der Instance-Aktualisierung fortzufahren.
- Max.: Legt den maximalen fehlerfreien Prozentsatz fest, der während der Instance-Aktualisierung möglich ist.

- iii. Erweitern Sie den Abschnitt Geschätzte temporäre Kapazität bei Austauscharbeiten auf der Grundlage der aktuellen Gruppengröße anzuzeigen, um zu überprüfen, ob die Werte für Min. und Max für Ihre Gruppe gelten. Welche genauen Werte verwendet werden, hängt vom gewünschten Kapazitätswert ab, der sich ändert, wenn die Gruppe skaliert wird.
- iv. Erweitern Sie den Abschnitt Fallback-Verhalten für ungültige Ersatzgrößen festlegen und wählen Sie dann aus, ob Sie gegen den maximalen fehlerfreien Prozentsatz verstoßen möchten, um der Verfügbarkeit Priorität einzuräumen, oder ob Sie gegen den minimalen fehlerfreien Prozentsatz verstoßen möchten.

Es wird für sehr kleine Gruppen nicht empfohlen, die Standardoption Minimaler fehlerfreie Prozentsatz beizubehalten. Wenn sich nur eine Instance in der Auto-Scaling-Gruppe befindet, kann das Starten einer Instance-Aktualisierung zu einem Ausfall führen.

Dieser Schritt konfiguriert das Fallback-Verhalten, wenn Sie eine Auto-Scaling-Gruppe verwenden, die noch keine Instance-Wartungsrichtlinie hat. Diese Option ist nicht verfügbar und wird nicht angezeigt, wenn Ihre Gruppe über eine Instance-Wartungsrichtlinie verfügt. Diese Option ist auch nur für die Ersatzmethode Beenden und Starten verfügbar. Bei anderen Ersatzmethoden wird die maximale Fehlerquote überschritten, um der Verfügbarkeit Priorität einzuräumen.

- b. Geben Sie für Instance-Aufwärmphase die Anzahl der Sekunden ein, ab der sich der Status einer neuen Instance in `InService` ändert, wenn sie ihre Initialisierung abgeschlossen hat. Amazon EC2 Auto Scaling wartet diese Zeit ab, bevor es mit dem Ersetzen der nächsten Instance fortfährt.

Während der Aufwärmphase wird eine neu gestartete Instance nicht zu den aggregierten Metriken der Auto-Scaling-Gruppe (z. B. `CPUUtilization`, `NetworkIn` und `NetworkOut`) gezählt. Wenn Sie der Auto-Scaling-Gruppe Skalierungsrichtlinien hinzugefügt haben, werden die Skalierungsaktivitäten parallel ausgeführt. Wenn Sie ein langes Intervall für die Aufwärmphase der Instance-Aktualisierung festlegen, dauert es länger, bis neu gestartete Instances in den Metriken angezeigt werden. Daher verhindert eine angemessene Aufwärmphase, dass Amazon EC2 Auto Scaling auf veraltete Metrikdaten skaliert.

Wenn Sie bereits eine standardmäßige Aufwärmphase für die Instance der Auto-Scaling-Gruppe definiert haben, müssen Sie die Aufwärmphase für die Instance nicht ändern. Wenn Sie die Standardeinstellung jedoch überschreiben möchten, können Sie einen Wert

für diese Option festlegen. Weitere Informationen zum Festlegen der standardmäßigen Aufwärmphase der Instance finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

5. Für Einstellungen aktualisieren nehmen Sie Folgendes vor:

- a. (Optional) Für Checkpoints (Prüfpunkte) wählen Sie Enable Checkpoints (Prüfpunkte aktivieren) aus, um Instances mit einem inkrementellen oder schrittweisen Ansatz für eine Instance-Aktualisierung zu ersetzen. Dies bietet zusätzliche Zeit für die Verifizierung verschiedener Ersatzvorgänge. Wenn Sie sich dafür entscheiden, Prüfpunkte nicht zu aktivieren, werden die Instances in einem fast kontinuierlichen Vorgang ersetzt.

Wenn Sie Prüfpunkte aktivieren, finden Sie unter [Aktivieren von Prüfpunkten \(Konsole\)](#) zusätzliche Schritte.

- b. Aktivieren oder Deaktivieren von Skip Matching (Abgleich überspringen):
 - Um das Ersetzen von Instances zu überspringen, die bereits mit Ihrer Startvorlage und allen Instance-Typ-Überschreibungen übereinstimmen, lassen Sie das Kontrollkästchen Überspringen des Abgleichs aktivieren aktiviert.
 - Wenn Sie das Überspringen des Abgleichs deaktivieren, indem Sie dieses Kontrollkästchen deaktivieren, können alle Instances ersetzt werden.

Wenn Sie das Überspringen des Abgleichs aktivieren, können Sie eine neue Startvorlage oder eine neue Version der Startvorlage festlegen, anstatt die vorhandene zu verwenden. Sie können dies im Abschnitt Gewünschte Konfiguration auf der Seite Instance-Aktualisierung starten tun. Sie können Ihre Instance-Typ-Überschreibungen auch in Gewünschte Konfiguration aktualisieren.

- c. Wählen Sie für Standby-Instances die Option Ignorieren, Beenden oder Warten aus. Dies legt fest, was passiert, wenn Instances im Standby-Status erkannt werden. Weitere Informationen finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#).

Wenn Sie Warten auswählen, müssen Sie zusätzliche Schritte unternehmen, um diese Instances wieder in Betrieb zu nehmen. Wenn Sie dies nicht tun, ersetzt die Instance-Aktualisierung alle InService-Instances und wartet eine Stunde. Wenn dann Standby-Instances übrig bleiben, schlägt die Instance-Aktualisierung fehl. Um diese Situation zu vermeiden, wählen Sie für diese Instances stattdessen Ignorieren oder Beenden aus.

- d. Wählen Sie für Vor Abskalierung geschützte Instances die Option Ignorieren, Ersetzen oder Warten aus. Dies legt fest, was passiert, wenn vor Abskalierung geschützte Instances erkannt werden. Weitere Informationen finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).

Wenn Sie Warten auswählen, müssen Sie zusätzliche Schritte unternehmen, um den Abskalierungsschutz dieser Instances zu entfernen. Wenn Sie dies nicht tun, ersetzt die Instance-Aktualisierung alle ungeschützten Instances und wartet eine Stunde. Wenn dann vor Abskalierung geschützte Instances übrig bleiben, schlägt die Instance-Aktualisierung fehl. Um diese Situation zu verhindern, wählen Sie stattdessen für diese Instances Ignorieren oder Ersetzen aus.

6. (Optional) Wählen Sie für CloudWatch Alarm die Option CloudWatch Alarme aktivieren und wählen Sie dann einen oder mehrere Alarme aus. CloudWatch Alarme können verwendet werden, um Probleme zu identifizieren und den Vorgang fehlschlagen zu lassen, wenn ein Alarm in den ALARM Status wechselt. Weitere Informationen finden Sie unter [Starten einer Instance-Aktualisierung mit automatischem Rollback](#).
7. Führen Sie im Abschnitt Gewünschte Konfiguration Folgendes aus.

Für diesen Schritt können Sie die JSON- oder YAML-Syntax verwenden, um Parameterwerte zu bearbeiten, anstatt eine Auswahl in der Konsolenschnittstelle zu treffen. Wählen Sie hierfür Use code editor (Code-Editor verwenden) anstelle von Use console interface (Konsolenschnittstelle verwenden) aus. Im folgenden Verfahren wird erläutert, wie mit der Konsolenschnittstelle eine Auswahl getroffen werden kann.


- a. Für Update launch template (Startvorlage aktualisieren):
 - Wenn Sie keine neue Startvorlage oder neue Startvorlagenversion für Ihre Auto-Scaling-Gruppe erstellt haben, aktivieren Sie dieses Kontrollkästchen nicht.
 - Wenn Sie eine neue Startvorlage oder eine neue Startvorlagen-Version erstellt haben, aktivieren Sie dieses Kontrollkästchen. Wenn Sie diese Option auswählen, zeigt Amazon EC2 Auto Scaling Ihnen die aktuelle Startvorlage und die aktuelle Version der Startvorlage an. Es listet auch alle anderen verfügbaren Versionen auf. Wählen Sie die Startvorlage und dann die Version aus.

Nachdem Sie eine Version ausgewählt haben, werden Ihnen die Versionsinformationen angezeigt. Dies ist die Version der Startvorlage, die beim Ersetzen von Instances im Rahmen einer Instance-Aktualisierung verwendet wird. Wenn die Instance-Aktualisierung

erfolgreich ist, wird diese Version der Startvorlage auch verwendet, wenn neue Instances gestartet werden, z. B. wenn die Gruppe skaliert wird.

- b. Für Use these settings to override the instance type and purchase option defined in the launch template (Verwenden Sie diese Einstellungen, um den Instance-Typ und die Kaufoption außer Kraft zu setzen, die in der Startvorlage definiert sind):

In der Standardeinstellung ist dieses Kontrollkästchen aktiviert. Amazon EC2 Auto Scaling füllt die einzelnen Parameter mit dem Wert, der derzeit in der Richtlinie für gemischte Instances für die Auto-Scaling-Gruppe festgelegt ist. Aktualisieren Sie nur die Werte für die Parameter, die Sie ändern möchten. Hinweise zu diesen Einstellungen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

 Warning

Es wird empfohlen, dieses Kontrollkästchen nicht zu deaktivieren. Deaktivieren Sie es nur, wenn Sie die Verwendung einer Richtlinie für gemischte Instances beenden möchten. Nach einer erfolgreichen Instance-Aktualisierung aktualisiert Amazon EC2 Auto Scaling Ihre Gruppe, damit Sie mit der gewünschten Konfiguration übereinstimmt. Wenn es keine Richtlinie für gemischte Instances mehr enthält, beendet Amazon EC2 Auto Scaling schrittweise alle derzeit ausgeführten Spot-Instances und ersetzt sie durch On-Demand-Instances. Oder, wenn Ihre Startvorlage Spot-Instances anfordert, beendet Amazon EC2 Auto Scaling schrittweise alle derzeit ausgeführten On-Demand-Instances und ersetzt sie durch Spot-Instances.

8. (Optional) Wählen Sie unter Rollback-Einstellungen die Option Automatisches Rollback aktivieren aus, um die Instance-Aktualisierung automatisch zurückzusetzen, falls sie fehlschlägt.

Diese Einstellung kann nur aktiviert werden, wenn die Auto-Scaling-Gruppe die Voraussetzungen für die Verwendung von Rollbacks erfüllt.

Weitere Informationen finden Sie unter [Änderungen mit manuellem oder auto Rollback rückgängig machen](#).

9. Überprüfen Sie Ihre gesamte Auswahl, um zu bestätigen, dass alles korrekt eingerichtet ist.

An dieser Stelle empfiehlt es sich sicherzustellen, dass sich die Unterschiede zwischen den aktuellen und den vorgeschlagenen Änderungen nicht auf unerwartete oder unerwünschte Weise

auf Ihre Anwendung auswirken. Informationen zur Bestätigung, dass Ihr Instance-Typ mit Ihrer Startvorlage kompatibel ist, finden Sie unter [Kompatibilität von Instance-Typen](#).

Wenn Sie mit der Auswahl für Ihre Instance-Aktualisierung zufrieden sind, klicken Sie auf Instance-Aktualisierung Starten.

Starten einer Instance-Aktualisierung (AWS CLI)

So starten Sie eine Instance-Aktualisierung

Verwenden Sie den folgenden [start-instance-refresh](#) Befehl, um eine Instanzaktualisierung von der aus zu starten AWS CLI. Sie können alle Voreinstellungen angeben, die Sie in einer JSON-Konfigurationsdatei ändern möchten. Wenn Sie auf die Konfigurationsdatei verweisen, geben Sie den Dateipfad und -namen an, wie im folgenden Beispiel gezeigt.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von config.json:

```
{
  "AutoScalingGroupName": "my-asg",
  "Preferences": {
    "InstanceWarmup": 60,
    "MinHealthyPercentage": 50,
    "AutoRollback": true,
    "ScaleInProtectedInstances": Ignore,
    "StandbyInstances": Terminate
  }
}
```

Wenn keine Voreinstellungen angegeben werden, werden die Standardwerte verwendet. Weitere Informationen finden Sie unter [Die Standardwerte für eine Instance-Aktualisierung verstehen](#).

Beispielausgabe:

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

Überwachen Sie eine Instanzaktualisierung mit dem AWS-Managementkonsole oder AWS CLI

Sie können eine laufende Instanzaktualisierung überwachen oder den Status vergangener Instanzaktualisierungen der letzten sechs Wochen mithilfe von AWS-Managementkonsole oder AWS CLI nachschlagen.

Überwachen und überprüfen Sie den Status einer Instanzaktualisierung

Verwenden Sie eine der folgenden Methoden, um den Status einer Instanzaktualisierung zu überwachen und zu überprüfen:

Console

Tip

In diesem Verfahren sollten die benannten Spalten bereits angezeigt werden. Um ausgeblendete Spalten anzuzeigen oder die Anzahl der angezeigten Zeilen zu ändern, wählen Sie das Zahnradsymbol in der oberen rechten Ecke des Abschnitts, um das Einstellungsfenster zu öffnen. Aktualisieren Sie die Einstellungen nach Bedarf und wählen Sie Bestätigen aus.

Um den Status einer Instanzaktualisierung zu überwachen und zu überprüfen (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Auf der Registerkarte Instance refresh (Instance-Aktualisierung) unter Instance refresh history (Instance-Aktualisierungsverlauf) können Sie den Status Ihrer Anforderung bestimmen, indem Sie sich die Spalte Status ansehen. Der Vorgang wechselt während der Initialisierung in Pending den Status. Der Status sollte sich dann schnell in InProgress ändern. Wenn alle Instances aktualisiert sind, ändert sich der Status in Successful.
4. Sie können den Erfolg oder Misserfolg laufender Aktivitäten weiter überwachen, indem Sie sich die Skalierungsaktivitäten der Gruppe ansehen. Auf der Registerkarte Activity (Aktivität) unter Activity history (Aktivitätsverlauf) werden beim Start der Instance-Aktualisierung

Einträge angezeigt, wenn Instances beendet werden. Beim Starten von Instances werden weitere Einträge angezeigt. Wenn Sie zahlreiche Skalierungsaktivitäten haben, können Sie sich mehr davon anzeigen lassen, indem Sie oben im Aktivitätsverlauf auf das Symbol > klicken. Informationen zur Behebung von Problemen, die zum Fehlschlagen von Aktivitäten führen können, finden Sie unter [Probleme in Amazon EC2 Auto Scaling beheben](#).

5. (Optional) Auf der Registerkarte Instanzverwaltung unter Instances können Sie bei Bedarf den Fortschritt bestimmter Instanzen überprüfen.

AWS CLI

Um den Status einer Instanzaktualisierung zu überwachen und zu überprüfen (AWS CLI)

Verwenden Sie den folgenden [describe-instance-refreshes](#)-Befehl.

```
aws autoscaling describe-instance-refreshes --auto-scaling-group-name my-asg
```

Es folgt eine Beispielausgabe.

Instanzaktualisierungen werden nach der Startzeit sortiert. Instanzaktualisierungen, die noch im Gange sind, werden zuerst beschrieben.

```
{
  "InstanceRefreshes": [
    {
      "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b",
      "AutoScalingGroupName": "my-asg",
      "Status": "InProgress",
      "StatusReason": "Waiting for instances to warm up before continuing. For example: i-0645704820a8e83ff is warming up.",
      "StartTime": "2023-11-24T16:46:52+00:00",
      "PercentageComplete": 50,
      "InstancesToUpdate": 0,
      "Preferences": {
        "MaxHealthyPercentage": 120,
        "MinHealthyPercentage": 90,
        "InstanceWarmup": 60,
        "SkipMatching": false,
        "AutoRollback": true,
        "ScaleInProtectedInstances": "Ignore",
        "StandbyInstances": "Ignore"
      }
    }
  ]
}
```

```

    }
  },
  {
    "InstanceRefreshId":"0e151305-1e57-4a32-a256-1fd14157c5ec",
    "AutoScalingGroupName":"my-asg",
    "Status":"Successful",
    "StartTime":"2023-11-22T13:53:37+00:00",
    "EndTime":"2023-11-22T13:59:45+00:00",
    "PercentageComplete":100,
    "InstancesToUpdate":0,
    "Preferences":{
      "MaxHealthyPercentage":120,
      "MinHealthyPercentage":90,
      "InstanceWarmup":60,
      "SkipMatching":false,
      "AutoRollback":true,
      "ScaleInProtectedInstances":"Ignore",
      "StandbyInstances":"Ignore"
    }
  }
]
}

```

Sie können den Erfolg oder Misserfolg laufender Aktivitäten weiter überwachen, indem Sie sich die Skalierungsaktivitäten der Gruppe ansehen. Die Skalierungsaktivitäten helfen Ihnen auch dabei, weitere Details zu finden, um Probleme bei einer Instance-Aktualisierung zu beheben. Weitere Informationen finden Sie unter [Probleme in Amazon EC2 Auto Scaling beheben](#).

Status von Instance-Aktualisierungen

Wenn Sie eine Instance-Aktualisierung starten, wechselt diese in den Status Ausstehend. Sie wechselt von „Ausstehend“ InProgress zu „Erfolgreich“, „Fehlgeschlagen“, „Storniert“ oder RollbackFailed. RollbackSuccessful

Eine Instance-Aktualisierung kann die folgenden Status haben:

Status	Description
Ausstehend	Die Anfrage wurde erstellt, aber die Instance-Aktualisierung wurde nicht gestartet.

Status	Description
InProgress	Eine Instance-Aktualisierung ist in Bearbeitung.
Erfolgreich	Eine Instance-Aktualisierung wurde erfolgreich abgeschlossen.
Fehlgeschlagen	Eine Instance-Aktualisierung konnte nicht abgeschlossen werden. Sie können Probleme mit dem Statusgrund und den Skalierungsaktivitäten beheben.
Abbrechen	Eine laufende Instance-Aktualisierung wird abgebrochen.
Abgebrochen	Die Instance-Aktualisierung wird abgebrochen.
RollbackInProgress	Eine Instance-Aktualisierung wird rückgängig gemacht.
RollbackFailed	Das Rollback konnte nicht abgeschlossen werden. Sie können Probleme mit dem Statusgrund und den Skalierungsaktivitäten beheben.
RollbackSuccessful	Das Rollback wurde erfolgreich abgeschlossen.
Backen	Das Warten auf die angegebene Backzeit nach einer Instanzaktualisierung hat die Aktualisierung der Instanzen abgeschlossen.

Ersetzen Sie die Root-Volumes während der Instanzaktualisierung

Inhalt

- [Funktionsweise](#)
- [Voraussetzungen](#)
- [Starten Sie eine Instanzaktualisierung mit Replace Root Volume](#)
- [Einschränkungen](#)
- [Verwenden Sie Lifecycle-Hooks mit Replace Root-Volume](#)

Funktionsweise

Durch den Austausch von Root-Volumes werden Ihre Instances aktualisiert, indem nur das Root-EBS-Volume ersetzt wird, während die Instance weiterhin läuft. Dadurch entfällt die Notwendigkeit,

neue Instances zu starten, und potenzielle Kapazitätsengpässe werden vermieden. Durch diesen Prozess werden alle anderen Instance-Ressourcen geschont, einschließlich:

- Netzwerkschnittstellen und IP-Adressen
- EBS-Volumes ohne Root-Rechte
- Instance-Speicher für Volumes und Daten
- Sicherheitsgruppen und IAM-Rollen

Während des Austauschs des Root-Volumes wird Ihre Anwendung weiterhin auf der vorhandenen Instance ausgeführt. Das ursprüngliche Root-Volume wird getrennt, ein neues Root-Volume wird aus Ihrem angegebenen AMI erstellt und dann an dieselbe Instance angehängt. Die ursprünglichen Root-Volumes werden nach erfolgreichem Austausch automatisch gelöscht.

Wenn Sie die Strategie zum Ersetzen von Root-Volumes verwenden, durchlaufen Instances die folgenden Lebenszyklusstatus:

1. `ReplacingRootVolume`- Der Austausch des Root-Volumes beginnt
2. `ReplacingRootVolume:Wait`- Wartet auf den Abschluss des Lifecycle-Hooks (falls konfiguriert)
3. `ReplacingRootVolume:Proceed`- Wird mit dem Austausch fortgefahren
4. `RootVolumeReplaced`- Der Austausch wurde erfolgreich abgeschlossen

Nachdem der Austausch des Root-Volumes abgeschlossen ist, führt Auto Scaling Integritätsprüfungen für aktualisierte Instances durch. Wenn der Austausch des Root-Volumes für eine Instance fehlschlägt, markiert Auto Scaling diese Instance zur Kündigung und ersetzt sie durch eine neue Instance.

Voraussetzungen

- Ihre Auto Scaling Scaling-Gruppe muss eine Richtlinie für gemischte Instanzen verwenden
- Für alle Überschreibungen in der Richtlinie für gemischte Instanzen muss Folgendes angegeben werden `ImageId`
- AMIs darf nur ein einziges Root-Volume enthalten
- Alle Instances müssen der Konfiguration der Startvorlage der Gruppe entsprechen
- Sie müssen die Instanzaktualisierung mit einer gewünschten Konfiguration starten, die eine Richtlinie für gemischte Instanzen enthält, die `ImageId` Überschreibungen enthält.

Starten Sie eine Instanzaktualisierung mit Replace Root Volume

Um eine Instanzaktualisierung mit Replace Root Volume (AWS CLI) zu starten

Verwenden Sie den folgenden [start-instance-refresh](#) Befehl, um eine Instanzaktualisierung von der aus zu starten AWS CLI. Sie können alle Voreinstellungen angeben, die Sie in einer JSON-Konfigurationsdatei ändern möchten. Wenn Sie auf die Konfigurationsdatei verweisen, geben Sie den Dateipfad und -namen an, wie im folgenden Beispiel gezeigt.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von config.json:

Example

```
{
  "AutoScalingGroupName": "my-asg",
  "Strategy" : "ReplaceRootVolume",
  "DesiredConfiguration":{
    "MixedInstancesPolicy":{
      "LaunchTemplate": {
        "LaunchTemplateSpecification": {
          "LaunchTemplateId": "my-launch-template",
          "Version": "1"
        },
      },
      "Overrides": [
        {
          "InstanceType": "c5.large",
          "ImageId": "ami-1234example"
        },
        {
          "InstanceType": "m5.large",
          "ImageId": "ami-2345example"
        }
      ]
    }
  },
  "Preferences": {
    "InstanceWarmup": 60,
    "MinHealthyPercentage": 90,
    "AutoRollback": true,
  },
}
```

```
"ScaleInProtectedInstances": Ignore,  
"StandbyInstances": Ignore  
}  
}
```

Wenn keine Voreinstellungen angegeben werden, werden die Standardwerte verwendet. Weitere Informationen finden Sie unter [Die Standardwerte für eine Instance-Aktualisierung verstehen](#).

Beispielausgabe:

```
{  
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

Einschränkungen

- Sie können eine Instance-Aktualisierung nicht mit Replace Root Volume starten, wenn die EC2 Auto Scaling Scaling-Gruppe oder die gewünschte Konfiguration für die Instance-Aktualisierung die Vorlagenversion verwendet `$Latest` oder die Vorlagenversion `$Default` startet.
- Sie können eine Instance-Aktualisierung nicht mit Replace Root Volume auf einer Amazon EC2 Auto Scaling Scaling-Gruppe starten, wenn sie Instances enthält, deren Instance-Typ nicht in der Richtlinie für gemischte Instanzen enthalten ist.
- Während einer Instance-Aktualisierung mit Replace Root Volume können Sie nur Instances mit Typen anhängen, die in der Richtlinie für gemischte Instanzen der gewünschten Konfiguration enthalten sind.
- Sie können eine Instance-Aktualisierung nicht mit Replace Root Volume auf einer EC2 Auto Scaling Scaling-Gruppe starten, die über einen warmen Pool verfügt.
- Sie können einer Amazon EC2 Auto Scaling Scaling-Gruppe, die über eine aktive Instance-Aktualisierung verfügt, keinen warmen Pool mit Replace Root Volume hinzufügen.

Verwenden Sie Lifecycle-Hooks mit Replace Root-Volume

Instances, bei denen das Root-Volume ersetzt wird, folgen ihren eigenen Lebenszyklusübergängen, sodass Sie Aktionen (z. B. eine Lambda-Funktion) vor und and/or nach dem Austausch aufrufen können. Weitere Informationen zu den Lebenszyklusstatus beim Austausch des Root-Volumes finden Sie unter [Statusübergänge im Lebenszyklus von Instances, bei denen das Root-Volume ausgetauscht wird](#)

Weitere Informationen über das Hinzufügen von Lebenszyklus-Hooks finden Sie unter [Fügen Sie Lifecycle-Hooks zu Ihrer Auto Scaling Scaling-Gruppe hinzu](#). Weitere Informationen über das Abschließen einer Lebenszyklus-Aktion finden Sie unter [Eine Lebenszyklusaktion in einer Auto Scaling Scaling-Gruppe abschließen](#).

Vor dem Austausch eines Stammvolumens kann ein Lifecycle-Hook in den folgenden Szenarien nützlich sein:

- Sie möchten Ihre Anwendung ordnungsgemäß herunterfahren, bevor das Root-Volume ersetzt und die Instance neu gestartet wird.
- Sie möchten Daten vom Root-Volume an einen anderen Speicherort verschieben, bevor das Volume ersetzt wird.

Nach dem Austausch des Root-Volumens kann ein Lifecycle-Hook in den folgenden Szenarien nützlich sein:

- Sie möchten sicherstellen, dass Ihre Instance vollständig für das neue AMI bereit ist, bevor sie Traffic empfängt.
- Sie möchten Anwendungsdaten, die zuvor auf dem Root-Volume vorhanden waren, bootstrappen.

Berücksichtigen Sie beim Hinzufügen von Lebenszyklus-Hooks Folgendes:

- Wenn ein Lifecycle-Hook für die `autoscaling:EC2_INSTANCE_TERMINATING` Lifecycle-Aktion konfiguriert ist, wird eine Instance, die kurz vor dem Austausch des Root-Volumens steht, pausiert, um eine benutzerdefinierte Aktion auszuführen, sobald sie den `ReplacingRootVolume:Wait` Status erreicht.
- Wenn ein Lifecycle-Hook für die `autoscaling:EC2_INSTANCE_LAUNCHING` Lifecycle-Aktion konfiguriert ist, wird eine Instance, deren Root-Volume gerade ersetzt wurde, pausiert, um eine benutzerdefinierte Aktion auszuführen, wenn sie den `Pending:Wait` Status erreicht.

Wenn Instances einen Wartezustand erreichen, sendet Amazon EC2 Auto Scaling eine Benachrichtigung. Die Lifecycle-Hook-Benachrichtigungen von Auto Scaling enthalten ein `Action` Feld mit einem Wert `ReplaceRootVolume`, der angibt, dass für die Instance ein Root-Volume ersetzt wird.

Beispiele für diese Benachrichtigungen finden Sie im EventBridge Abschnitt dieses Handbuchs. Weitere Informationen finden Sie unter [Die Aktualisierung der Instanz ersetzt Ereignisse im Lebenszyklus des Root-Volumes](#).

Brechen Sie eine Instanzaktualisierung mit dem AWS-Managementkonsole oder ab AWS CLI

Sie können eine Instance-Aktualisierung abbrechen, die noch ausgeführt wird. Sie können den Vorgang nicht mehr abbrechen, nachdem er beendet ist.

Das Abbrechen einer Instance-Aktualisierung setzt keine Instances zurück, die bereits ersetzt wurden. Führen Sie stattdessen ein Rollback durch, um die Änderungen an Ihren Instances rückgängig zu machen. Weitere Informationen finden Sie unter [Änderungen mit manuellem oder auto Rollback rückgängig machen](#).

Themen

- [Abbrechen einer Instance-Aktualisierung \(Konsole\)](#)
- [Abbrechen einer Instance-Aktualisierung \(AWS CLI\)](#)

Abbrechen einer Instance-Aktualisierung (Konsole)

So brechen Sie eine Instance-Aktualisierung ab

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.
3. Wählen Sie auf der Registerkarte Instance-Aktualisierung unter Aktive Instance-Aktualisierung die Option Aktionen und dann Abbrechen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Confirm (Bestätigen).

Der Status der Instance-Aktualisierung ist auf Abbrechen gesetzt. Nach Abschluss des Abbruchs wird der Status der Instance-Aktualisierung auf Abgebrochen gesetzt.

Abbrechen einer Instance-Aktualisierung (AWS CLI)

So brechen Sie eine Instance-Aktualisierung ab

Verwenden Sie den [cancel-instance-refresh](#) Befehl von AWS CLI und geben Sie den Namen der Auto Scaling Group ein.

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg
```

Beispielausgabe:

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

Um eine Instanzaktualisierung abubrechen, ohne auf den Übergang von Instances zu warten

Verwenden Sie die `--no-wait-for-transitioning-instances` Option mit dem [cancel-instance-refresh](#) Befehl von, AWS CLI um eine Instanzaktualisierung abubrechen, ohne auf den Instanzwechsel zu warten.

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg --no-wait-for-transitioning-instances
```

Wenn Sie eine Instance-Aktualisierung stornieren, wartet Amazon EC2 Auto Scaling standardmäßig darauf, dass alle Starts und Beenden während des Fluges abgeschlossen sind, bevor der Status auf Storniert gesetzt wird. Wenn Sie Instances mit langen Ladezeiten haben oder sofort eine neue Instance-Aktualisierung starten müssen, können Sie den Befehl verwenden, damit Sie nicht auf diese Übergangsvorgänge warten müssen.

Wenn Sie diese Option verwenden, ändert sich der Status der Instanzaktualisierung in Storniert, sodass Sie eine neue Instanzaktualisierung starten können. Alle Starts und Abbrüche während des Fluges werden im Hintergrund fortgesetzt.

Änderungen mit manuellem oder auto Rollback rückgängig machen

Sie können eine Instance-Aktualisierung, die noch ausgeführt wird, rückgängig machen. Sie können den Vorgang nicht mehr rückgängig machen, nachdem er beendet ist. Sie können Ihre Auto-Scaling-Gruppe jedoch erneut aktualisieren, indem Sie eine neue Instance-Aktualisierung starten.

Beim Rollback ersetzt Amazon EC2 Auto Scaling die Instances, die bisher bereitgestellt wurden. Die neuen Instances entsprechen der letzten Konfiguration, die Sie in der Auto-Scaling-Gruppe gespeichert haben, bevor Sie mit der Instance-Aktualisierung begonnen haben.

Amazon EC2 Auto Scaling bietet die folgenden Möglichkeiten ein Rollback durchzuführen:

- **Manuelles Rollback:** Sie starten ein Rollback manuell, um das, was bis zum Rollback-Punkt bereitgestellt wurde, rückgängig zu machen.
- **Automatisches Rollback:** Amazon EC2 Auto Scaling macht automatisch rückgängig, was bereitgestellt wurde, wenn die Instance-Aktualisierung aus irgendeinem Grund fehlschlägt oder wenn von Ihnen angegebene CloudWatch Alarme in den Status wechseln. ALARM

Inhalt

- [Überlegungen](#)
- [Manuelles Starten eines Rollbacks](#)
- [Starten einer Instance-Aktualisierung mit automatischem Rollback](#)

Überlegungen

Die folgenden Überlegungen gelten für die Verwendung eines Rollbacks:

- Die Rollback-Option ist nur verfügbar, wenn Sie beim Starten einer Instance-Aktualisierung eine gewünschte Konfiguration angeben.
- Sie können nur dann zu einer früheren Version einer Startvorlage zurückkehren, wenn es sich bei der Version um eine bestimmte nummerierte Version handelt. Die Rollback-Option ist nicht verfügbar, wenn die Auto-Scaling-Gruppe so konfiguriert ist, dass sie die Startvorlagenversion `$Latest` oder `$Default` verwendet.
- Sie können auch nicht zu einer Startvorlage zurückkehren, die für die Verwendung eines AMI-Alias aus dem AWS Systems Manager Parameterspeicher konfiguriert ist.
- Die Konfiguration, die Sie zuletzt in der Auto-Scaling-Gruppe gespeichert haben, muss sich in einem stabilen Zustand befinden. Wenn er sich nicht in einem stabilen Zustand befindet, wird der Rollback-Workflow trotzdem ausgeführt, aber er wird letztendlich fehlschlagen. Bis Sie das Problem behoben haben, befindet sich die Auto-Scaling-Gruppe möglicherweise in einem fehlerhaften Status, in dem Instances nicht mehr erfolgreich gestartet werden können. Dies kann die Verfügbarkeit des Services oder der Anwendung beeinträchtigen.

Manuelles Starten eines Rollbacks

Console

So starten Sie manuell ein Rollback einer Instance-Aktualisierung (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.
3. Wählen Sie auf der Registerkarte Instance-Aktualisierung unter Aktive Instance-Aktualisierung die Optionen Aktionen und dann Rollback starten.
4. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Confirm (Bestätigen).

AWS CLI

So starten Sie manuell ein Rollback einer Instance-Aktualisierung (AWS CLI)

Verwenden Sie den [rollback-instance-refresh](#) Befehl von AWS CLI und geben Sie den Namen der Auto Scaling Scaling-Gruppe ein.

```
aws autoscaling rollback-instance-refresh --auto-scaling-group-name my-asg
```

Beispielausgabe:

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

Tip

Wenn dieser Befehl einen Fehler ausgibt, stellen Sie sicher, dass Sie die AWS CLI lokale Version auf die neueste Version aktualisiert haben.

Starten einer Instance-Aktualisierung mit automatischem Rollback

Mithilfe der auto Rollback-Funktion können Sie die Instance-Aktualisierung automatisch rückgängig machen, wenn sie fehlschlägt, z. B. wenn Fehler auftreten oder ein bestimmter CloudWatch Amazon-Alarm in den ALARM Status wechselt.

Wenn Sie das automatische Rollback aktivieren und beim Ersetzen von Instances Fehler auftreten, versucht die Instance-Aktualisierung eine Stunde lang, alle Ersetzungen abzuschließen, bevor sie fehlschlägt und ein Rollback erfolgt. Diese Fehler werden in der Regel dadurch verursacht, dass EC2-Startfehler, falsch konfigurierte Integritätsprüfungen oder das Nichtignorieren oder Beenden von Instances, die sich im Status Standby befinden oder vor dem Abskalieren geschützt sind, nicht ignoriert oder zugelassen werden.

Die Angabe von CloudWatch Alarmen ist optional. Um einen Alarm anzugeben, müssen Sie ihn zunächst erstellen. Sie können Metrieralarme und zusammengesetzte Alarme angeben. Informationen zum Erstellen des Alarms finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#). Wenn Sie beispielsweise Elastic Load Balancing-Metriken verwenden und einen Application Load Balancer verwenden, könnten Sie die Metriken HTTPCode_ELB_5XX_Count und HTTPCode_ELB_4XX_Count verwenden.

Überlegungen

- Wenn Sie einen CloudWatch Alarm angeben, aber kein auto Rollback aktivieren und der Alarmstatus auf wechseltALARM, schlägt die Instanzaktualisierung ohne Rollback fehl.
- Sie können maximal 10 Alarme auswählen, wenn Sie eine Instance-Aktualisierung starten.
- Bei der Auswahl eines CloudWatch Alarms muss sich der Alarm in einem kompatiblen Zustand befinden. Wenn der Alarmstatus INSUFFICIENT_DATA oder ALARM ist, erhalten Sie eine Fehlermeldung, wenn Sie versuchen, die Instance-Aktualisierung zu starten.
- Wenn Sie einen Alarm für Amazon EC2 Auto Scaling erstellen, sollte der Alarm beinhalten, wie mit fehlenden Datenpunkten umzugehen ist. Wenn bei einer Metrik planmäßig häufig Datenpunkte fehlen, ist der Status des Alarms während dieser Zeiträume INSUFFICIENT_DATA. In diesem Fall kann Amazon EC2 Auto Scaling keine Instances ersetzen, bis neue Datenpunkte gefunden werden. Um den Alarm zu zwingen, den vorherigen Zustand ALARM oder OK beizubehalten, können Sie stattdessen fehlende Daten ignorieren. Weitere Informationen finden Sie unter [Konfiguration der Behandlung fehlender Daten durch Alarme](#) im CloudWatch Amazon-Benutzerhandbuch.

Console

So starten Sie eine Instance-Aktualisierung mit automatischem Rollback (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.
3. Wählen Sie auf der Registerkarte Instance refresh (Instance-Aktualisierung) unter Active instance refresh (Aktive Instance-Aktualisierung) die Option Start instance refresh (Instance-Aktualisierung starten) aus.
4. Befolgen Sie das [Starten einer Instance-Aktualisierung \(Konsole\)](#)-Verfahren und konfigurieren Sie die Einstellungen Ihrer Instance-Aktualisierungen nach Bedarf.
5. (Optional) Wählen Sie unter Einstellungen aktualisieren für CloudWatch Alarm die Option CloudWatch Alarme aktivieren und wählen Sie dann einen oder mehrere Alarme aus, um Probleme zu identifizieren und den Vorgang fehlschlagen zu lassen, wenn ein Alarm in den ALARM Status wechselt.
6. Wählen Sie unter Rollback-Einstellungen die Option Automatisches Rollback aus, um eine fehlgeschlagene Instance-Aktualisierung automatisch auf die Konfiguration zurückzusetzen, die Sie zuletzt in der Auto-Scaling-Gruppe gespeichert haben, bevor Sie mit der Instance-Aktualisierung begonnen haben.
7. Überprüfen Sie Ihre Auswahl und wählen Sie dann Instance-Aktualisierung starten.

AWS CLI

So starten Sie eine Instance-Aktualisierung mit automatischem Rollback (AWS CLI)

Verwenden Sie den [start-instance-refresh](#) Befehl und geben Sie `true` die `AutoRollback` Option in der `anPreferences`.

Das folgende Beispiel zeigt, wie eine Instance-Aktualisierung gestartet wird, die automatisch zurückgesetzt wird, wenn etwas fehlschlägt. Ersetzen Sie die *italicized* Parameterwerte durch eigene.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von `config.json`.

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1"
    }
  },
  "Preferences": {
    "AutoRollback": true
  }
}
```

Sie können auch ein automatisches Rollback durchführen, wenn die Instanzaktualisierung fehlschlägt oder wenn sich ein bestimmter CloudWatch Alarm im ALARM Status befindet, indem Sie die `AlarmSpecification` Option in Preferences und den Namen des Alarms angeben, wie im folgenden Beispiel. Ersetzen Sie die *italicized* Parameterwerte durch eigene.

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1"
    }
  },
  "Preferences": {
    "AutoRollback": true,
    "AlarmSpecification": { "Alarms": [ "my-alarm" ] }
  }
}
```

Bei erfolgreicher Ausführung gibt der Befehl eine Ausgabe zurück, die in etwa wie folgt aussieht:

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

 Tip

Wenn dieser Befehl einen Fehler auslöst, stellen Sie sicher, dass Sie die AWS CLI lokale Version auf die neueste Version aktualisiert haben.

Verwenden einer Instance-Aktualisierung mit Funktion zum Überspringen des Abgleichs

Das Überspringen des Abgleichs weist Amazon EC2 Auto Scaling an, Instances zu ignorieren, die bereits über Ihre neuesten Aktualisierungen verfügen. Auf diese Weise ersetzen Sie nicht mehr Instances als Sie benötigen. Dies ist hilfreich, wenn Sie sicherstellen möchten, dass Ihre Auto-Scaling-Gruppe eine bestimmte Version Ihrer Startvorlage verwendet und nur die Instances ersetzt, die eine andere Version verwenden.

Beachten Sie im Zusammenhang mit der Funktion zum Überspringen entsprechender Instances folgende Überlegungen:

- Wenn Sie eine Instance-Aktualisierung sowohl mit der Funktion zum Überspringen des Abgleichs als auch mit einer gewünschten Konfiguration starten, prüft Amazon EC2 Auto Scaling, ob Instances Ihrer gewünschten Konfiguration entsprechen. Anschließend werden nur die Instances ersetzt, die nicht Ihrer gewünschten Konfiguration entsprechen. Nach einer erfolgreichen Instance-Aktualisierung aktualisiert Amazon EC2 Auto Scaling die Gruppe, damit sie mit Ihrer gewünschten Konfiguration übereinstimmt.
- Wenn Sie eine Instance-Aktualisierung mit der Funktion zum Überspringen des Abgleichs starten, aber keine gewünschte Konfiguration angeben, prüft Amazon EC2 Auto Scaling, ob Instances mit der exakten Konfiguration übereinstimmen, die Sie zuletzt in der Auto-Scaling-Gruppe gespeichert haben. Anschließend ersetzt es nur die Instances, die nicht Ihrer zuletzt gespeicherten Konfiguration entsprechen.
- Sie können das Überspringen des Abgleichs mit einer neuen Startvorlage, einer neuen Version einer Startvorlage oder einem Satz von Instance-Typen verwenden. Wenn Sie das Überspringen des Abgleichs aktivieren, aber alles unverändert bleibt, ist die Instance-Aktualisierung sofort erfolgreich, ohne dass Instances ersetzt werden. Wenn Sie weitere Änderungen an Ihrer gewünschten Konfiguration vorgenommen haben (z. B. Änderungen an der Spot-Zuordnungsstrategie), wartet die Lösung Amazon EC2 Auto Scaling, bis die Instance-Aktualisierung erfolgreich ausgeführt wurde. Anschließend werden die Einstellungen der Auto-Scaling-Gruppe aktualisiert, sodass sie der neuen gewünschten Konfiguration entsprechen.

- Die Funktion zum Überspringen entsprechender Instances kann nicht mit einer neuen Startkonfiguration verwendet werden.
- Wenn Sie eine Instance-Aktualisierung starten und eine gewünschte Konfiguration angeben, stellt Amazon EC2 Auto Scaling sicher, dass alle Instances Ihre gewünschte Konfiguration verwenden. Wenn Sie also entweder `$Default` oder `$Latest` als gewünschte Version für Ihre Startvorlage angeben und dann während einer Instance-Aktualisierung eine neue Version der Startvorlage erstellen, werden alle Instances, die bereits ersetzt wurden, erneut ersetzt.
- Bei Skip Matching wird nicht ermittelt, ob ein Benutzerdatenskript in der Startvorlage aktualisierten Code abrufen und ihn auf neuen Instances installiert. Aus diesem Grund überspringt Skip Matching möglicherweise das Ersetzen von Instanzen, auf denen veralteter Code installiert ist. In diesem Fall sollten Sie den Abgleich überspringen deaktivieren, um sicherzustellen, dass alle Instances Ihren neuesten Code erhalten, auch ohne ein Versionsupdate für die Startvorlage.

Dieser Abschnitt enthält AWS CLI Anweisungen zum Starten einer Instanzaktualisierung bei aktiviertem Skip-Matching. Weitere Informationen zur Verwendung der Konsole finden Sie unter [Starten einer Instance-Aktualisierung \(Konsole\)](#).

Überspringen des Abgleichs (einfaches Verfahren)

Folgen Sie den Schritten in diesem Abschnitt AWS CLI , um Folgendes zu tun:

- Erstellen einer Startvorlage, die Sie auf Ihre Instances anwenden möchten.
- Starten einer Instance-Aktualisierung, um Ihre Startvorlage auf Ihre Auto-Scaling-Gruppe anzuwenden. Wenn Sie das Überspringen des Abgleichs nicht aktivieren, werden alle Instances ersetzt. Dies gilt auch dann, wenn die Startvorlage, die zum Bereitstellen der Instance verwendet wird, dieselbe ist wie die, die Sie für Ihre gewünschte Konfiguration angegeben haben.

So verwenden Sie die Funktion zum Überspringen des Abgleichs mit einer neuen Startvorlage

1. Verwenden Sie den [create-launch-template](#) Befehl, um eine neue Startvorlage für Ihre Auto Scaling Gruppe zu erstellen. Schließen Sie die `--launch-template-data`-Option und die JSON-Eingabe ein, die die Details der Instances definiert, die für Ihre Auto-Scaling-Gruppe erstellt werden.

Verwenden Sie beispielsweise den folgenden Befehl, um eine grundlegende Startvorlage mit der AMI-ID `ami-0123456789abcdef0` und dem Instance-Typ `t2.micro` zu erstellen.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
--version-description version1 \
--launch-template-data
'{"ImageId": "ami-0123456789abcdef0", "InstanceType": "t2.micro"}'
```

Bei erfolgreicher Ausführung gibt der Befehl eine Ausgabe zurück, die in etwa wie folgt aussieht:

```
{
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-068f72b729example",
    "LaunchTemplateName": "my-template-for-auto-scaling",
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "CreateTime": "2023-01-30T18:16:06.000Z",
    "DefaultVersionNumber": 1,
    "LatestVersionNumber": 1
  }
}
```

Weitere Informationen finden Sie unter [Beispiele für die Erstellung und Verwaltung von Startvorlagen mit dem AWS CLI](#).

2. Verwenden Sie den [start-instance-refresh](#) Befehl, um den Instance-Ersatz-Workflow zu initiieren und Ihre neue Startvorlage mit der ID anzuwenden *lt-068f72b729example*. Da die Startvorlage neu ist, verfügt sie nur über eine Version. Dies bedeutet, dass die Version 1 der Startvorlage das Ziel dieser Instance-Aktualisierung ist. Falls während der Instance-Aktualisierung ein Aufskalierungsereignis auftritt und Amazon EC2 Auto Scaling neue Instances mit der Version 1 dieser Startvorlage bereitstellt, werden diese nicht ersetzt. Nach erfolgreichem Abschluss des Vorgangs wird die neue Startvorlage erfolgreich auf Ihre Auto-Scaling-Gruppe angewendet.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von `config.json`.

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "LaunchTemplate": {
      "LaunchTemplateId": "lt-068f72b729example",
```

```
    "Version": "$Default"  
  }  
},  
"Preferences": {  
  "SkipMatching": true  
}  
}
```

Bei erfolgreicher Ausführung gibt der Befehl eine Ausgabe zurück, die in etwa wie folgt aussieht:

```
{  
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

Überspringen des Abgleichs (Gruppe mit gemischten Instances)

Wenn Sie eine Auto Scaling Scaling-Gruppe mit einer [Richtlinie für gemischte Instanzen](#) haben, folgen Sie den Schritten in diesem Abschnitt, um eine Instanzaktualisierung mit Skip-Matching AWS CLI zu starten. Ihnen stehen folgende Optionen zur Verfügung:

- Stellen Sie eine neue Startvorlage bereit, die für alle in der Richtlinie angegebenen Instance-Typen gelten soll.
- Stellen Sie einen aktualisierten Satz von Instance-Typen mit oder ohne Änderung der Startvorlage in der Richtlinie bereit. Beispielsweise möchten Sie möglicherweise von unerwünschten Instance-Typen weg migrieren. Sie würden die Startvorlage unverändert verwenden, ohne das AMI, die Sicherheitsgruppen oder andere Besonderheiten der zu ersetzenden Instances zu ändern.

Befolgen Sie die Schritte in einem der folgenden Abschnitte, je nachdem, welche Option Ihren Anforderungen entspricht.

So verwenden Sie die Funktion zum Überspringen des Abgleichs mit einer neuen Startvorlage

1. Verwenden Sie den [create-launch-template](#)Befehl, um eine neue Startvorlage für Ihre Auto Scaling Scaling-Gruppe zu erstellen. Schließen Sie die `--launch-template-data`-Option und die JSON-Eingabe ein, die die Details der Instances definiert, die für Ihre Auto-Scaling-Gruppe erstellt werden.

Verwenden Sie beispielsweise den folgenden Befehl, um eine Startvorlage mit der AMI-ID *ami-0123456789abcdef0* zu erstellen.

```
aws ec2 create-launch-template --launch-template-name my-new-template --version-  
description version1 \  
--launch-template-data '{"ImageId":"ami-0123456789abcdef0"}'
```

Bei erfolgreicher Ausführung gibt der Befehl eine Ausgabe zurück, die in etwa wie folgt aussieht:

```
{  
  "LaunchTemplate": {  
    "LaunchTemplateId": "lt-04d5cc9b88example",  
    "LaunchTemplateName": "my-new-template",  
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",  
    "CreateTime": "2023-01-31T15:56:02.000Z",  
    "DefaultVersionNumber": 1,  
    "LatestVersionNumber": 1  
  }  
}
```

Weitere Informationen finden Sie unter [Beispiele für die Erstellung und Verwaltung von Startvorlagen mit dem AWS CLI](#).

2. Führen Sie den [describe-auto-scaling-groups](#)Befehl aus, um die bestehende Richtlinie für gemischte Instanzen für Ihre Auto Scaling Scaling-Gruppe anzuzeigen. Sie benötigen diese Informationen im nächsten Schritt, wenn Sie die Instance-Aktualisierung starten.

Der folgende Beispielbefehl gibt die Richtlinie für gemischte Instances zurück, die für die benannte Auto-Scaling-Gruppe konfiguriert ist *my-asg*.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Bei erfolgreicher Ausführung gibt der Befehl eine Ausgabe zurück, die in etwa wie folgt aussieht:

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "AutoScalingGroupARN": "arn",  
      "MixedInstancesPolicy": {
```

```

    "LaunchTemplate":{
      "LaunchTemplateSpecification":{
        "LaunchTemplateId":"lt-073693ed27example",
        "LaunchTemplateName":"my-old-template",
        "Version":"$Default"
      },
      "Overrides":[
        {
          "InstanceType":"c5.large"
        },
        {
          "InstanceType":"c5a.large"
        },
        {
          "InstanceType":"m5.large"
        },
        {
          "InstanceType":"m5a.large"
        }
      ]
    },
    "InstancesDistribution":{
      "OnDemandAllocationStrategy":"prioritized",
      "OnDemandBaseCapacity":1,
      "OnDemandPercentageAboveBaseCapacity":50,
      "SpotAllocationStrategy":"price-capacity-optimized"
    }
  },
  "MinSize":1,
  "MaxSize":5,
  "DesiredCapacity":4,
  ...
}
]
}

```

3. Verwenden Sie den [start-instance-refresh](#) Befehl, um den Workflow zum Ersetzen von Instances zu initiieren und Ihre neue Startvorlage mit der ID anzuwenden *lt-04d5cc9b88example*. Da die Startvorlage neu ist, verfügt sie nur über eine Version. Dies bedeutet, dass die Version 1 der Startvorlage das Ziel dieser Instance-Aktualisierung ist. Falls während der Instance-Aktualisierung ein Aufskalierungsereignis auftritt und Amazon EC2 Auto Scaling neue Instances mit der Version 1 dieser Startvorlage bereitstellt, werden diese nicht ersetzt. Nach erfolgreichem

Abschluss des Vorgangs wird die aktualisierte Richtlinie für gemischte Instances erfolgreich auf Ihre Auto-Scaling-Gruppe angewendet.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von config.json.

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "MixedInstancesPolicy": {
      "LaunchTemplate": {
        "LaunchTemplateSpecification": {
          "LaunchTemplateId": "lt-04d5cc9b88example",
          "Version": "$Default"
        },
        "Overrides": [
          {
            "InstanceType": "c5.large"
          },
          {
            "InstanceType": "c5a.large"
          },
          {
            "InstanceType": "m5.large"
          },
          {
            "InstanceType": "m5a.large"
          }
        ]
      },
      "InstancesDistribution": {
        "OnDemandAllocationStrategy": "prioritized",
        "OnDemandBaseCapacity": 1,
        "OnDemandPercentageAboveBaseCapacity": 50,
        "SpotAllocationStrategy": "price-capacity-optimized"
      }
    }
  },
  "Preferences": {
    "SkipMatching": true
  }
}
```

```
}  
}
```

Bei erfolgreicher Ausführung gibt der Befehl eine Ausgabe zurück, die in etwa wie folgt aussieht:

```
{  
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

In diesem nächsten Verfahren stellen Sie einen aktualisierten Satz von Instance-Typen bereit, ohne die Startvorlage zu ändern.

So verwenden Sie die Funktion zum Überspringen des Abgleichs mit einem aktualisierten Satz von Instance-Typen

1. Führen Sie den [describe-auto-scaling-groups](#)Befehl aus, um die bestehende Richtlinie für gemischte Instanzen für Ihre Auto Scaling Scaling-Gruppe anzuzeigen. Sie benötigen diese Informationen im nächsten Schritt, wenn Sie die Instance-Aktualisierung starten.

Der folgende Beispielbefehl gibt die Richtlinie für gemischte Instances zurück, die für die benannte Auto-Scaling-Gruppe konfiguriert ist *my-asg*.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Bei erfolgreicher Ausführung gibt der Befehl eine Ausgabe zurück, die in etwa wie folgt aussieht:

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "AutoScalingGroupARN": "arn",  
      "MixedInstancesPolicy": {  
        "LaunchTemplate": {  
          "LaunchTemplateSpecification": {  
            "LaunchTemplateId": "lt-073693ed27example",  
            "LaunchTemplateName": "my-template-for-auto-scaling",  
            "Version": "$Default"  
          },  
          "Overrides": [  
            {  

```

```

        "InstanceType":"c5.large"
    },
    {
        "InstanceType":"c5a.large"
    },
    {
        "InstanceType":"m5.large"
    },
    {
        "InstanceType":"m5a.large"
    }
]
},
"InstancesDistribution":{
    "OnDemandAllocationStrategy":"prioritized",
    "OnDemandBaseCapacity":1,
    "OnDemandPercentageAboveBaseCapacity":50,
    "SpotAllocationStrategy":"price-capacity-optimized"
}
},
"MinSize":1,
"MaxSize":5,
"DesiredCapacity":4,
...
}
]
}

```

2. Verwenden Sie den [start-instance-refresh](#) Befehl, um den Workflow für den Instanzersatz zu initiieren und Ihre Updates anzuwenden. Wenn Sie Instances ersetzen möchten, die bestimmte Instance-Typen verwenden, muss Ihre gewünschte Konfiguration die Richtlinie für gemischte Instances nur mit den gewünschten Instance-Typen angeben. Sie können wählen, ob Sie stattdessen neue Instance-Typen hinzufügen möchten.

Der folgende Beispielbefehl startet eine Instance-Aktualisierung ohne den unerwünschten Instance-Typ *m5a.Large*. Wenn ein Instance-Typ in Ihrer Gruppe nicht mit einem der übrigen drei Instance-Typen übereinstimmt, werden die Instances ersetzt. (Berücksichtigen Sie, dass eine Instance-Aktualisierung nicht die Instance-Typen auswählt, aus denen die neuen Instances bereitgestellt werden sollen. Dies wird stattdessen von den [Zuweisungsstrategien](#) erledigt.) Nach erfolgreichem Abschluss des Vorgangs wird die aktualisierte Richtlinie für gemischte Instances erfolgreich auf Ihre Auto-Scaling-Gruppe angewendet.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von config.json

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "MixedInstancesPolicy": {
      "LaunchTemplate": {
        "LaunchTemplateSpecification": {
          "LaunchTemplateId": "lt-073693ed27example",
          "Version": "$Default"
        },
        "Overrides": [
          {
            "InstanceType": "c5.large"
          },
          {
            "InstanceType": "c5a.large"
          },
          {
            "InstanceType": "m5.large"
          }
        ]
      },
      "InstancesDistribution": {
        "OnDemandAllocationStrategy": "prioritized",
        "OnDemandBaseCapacity": 1,
        "OnDemandPercentageAboveBaseCapacity": 50,
        "SpotAllocationStrategy": "price-capacity-optimized"
      }
    }
  },
  "Preferences": {
    "SkipMatching": true
  }
}
```

Prüfpunkte zu einer Instance-Aktualisierung hinzufügen

Wenn Sie eine Instance-Aktualisierung verwenden, können Sie Instances phasenweise ersetzen, damit Sie bei laufendem Betrieb Überprüfungen für Ihre Instances durchführen können. Um eine schrittweise Ersetzung durchzuführen, fügen Sie Checkpoints hinzu. Dies sind Zeitpunkte, an denen die Instance-Aktualisierung pausiert wird. Die Verwendung von Prüfpunkten gibt Ihnen eine bessere Kontrolle darüber, wie Sie Ihre Auto-Scaling-Gruppe aktualisieren. Damit können Sie bestätigen, dass Ihre Anwendung zuverlässig und vorhersehbar funktioniert.

Inhalt

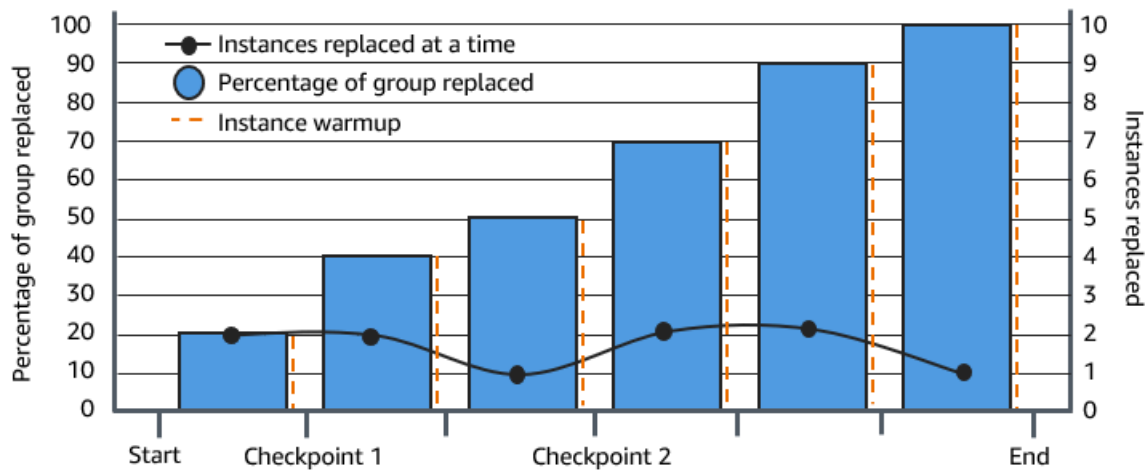
- [Funktionsweise](#)
- [Überlegungen](#)
- [Aktivieren Sie Checkpoints mithilfe von oder AWS-Managementkonsole AWS CLI](#)

Funktionsweise

Wenn Sie eine Instance-Aktualisierung starten, geben Sie Checkpoints als Prozentsätze der Gesamtzahl der Instances in der Auto Scaling Scaling-Gruppe an. Diese Checkpoints geben den Mindestprozentsatz der Instances in der Auto Scaling Scaling-Gruppe an, die neue Instances sein müssen, bevor der Checkpoint als erreicht gilt. Wenn Ihre Checkpoints beispielsweise so sind [20, 50, 100], ist der erste Checkpoint erreicht, wenn 20 Prozent der Instances neu sind, der zweite, wenn 50 Prozent neu sind, und der letzte Checkpoint, wenn alle Instances neu sind.

Amazon EC2 Auto Scaling passt Instance-Ersetzungen so an, dass sie die angegebenen Checkpoint-Prozentsätze einhalten und gleichzeitig den minimalen fehlerfreien Prozentsatz der Gruppe beibehalten. Um einen Checkpoint-Prozentsatz zu erreichen, ersetzt Amazon EC2 Auto Scaling manchmal weniger fest, aber nie mehr, als der minimale fehlerfreie Prozentsatz zulässt.

Stellen Sie sich die folgende Auto-Scaling-Gruppe mit 10 Instanzen vor. Die Prozentsätze der Kontrollpunkte sind [20, 50, 100], der minimale fehlerfreie Prozentsatz ist 80 Prozent, und der maximale fehlerfreie Prozentsatz ist 100 Prozent. Um den minimalen fehlerfreien Prozentsatz aufrechtzuerhalten, können nur zwei Instances auf einmal ersetzt werden. Im folgenden Diagramm ist der Prozess zum Ersetzen aller Instances dargestellt, bevor ein Checkpoint erreicht wird.



Im obigen Beispiel gibt es für jede neue Instance, die gestartet wird, eine Instance-Aufwärmphase. Möglicherweise haben Sie auch einen Lebenszyklus-Hook, der eine Instance in einen Wartestatus versetzt und dann eine benutzerdefinierte Aktion ausführt, während sie gestartet oder beendet wird.

Amazon EC2 Auto Scaling gibt Ereignisse für jeden Checkpoint aus, mit Ausnahme des Checkpoints, der zu 100 Prozent abgeschlossen ist. Sie können eine EventBridge Regel hinzufügen, um die Ereignisse an ein Ziel wie Amazon SNS zu senden. So werden Sie benachrichtigt, wenn Sie die erforderlichen Überprüfungen durchführen können. Weitere Informationen finden Sie unter [Erstellen Sie EventBridge Regeln für Instance-Aktualisierungsereignisse](#).

Überlegungen

Behalten Sie bei der Verwendung von Prüfpunkten die folgenden Überlegungen im Auge:

- Da Prüfpunkte auf Prozentsätzen basieren, ändert sich die Anzahl der zu ersetzenden Instances mit der Größe der Gruppe. Wenn eine Scale-Out-Aktivität stattfindet und die Größe der Gruppe zunimmt, kann ein laufender Vorgang erneut einen Checkpoint erreichen. In diesem Fall sendet Amazon EC2 Auto Scaling eine weitere Benachrichtigung und wiederholt die Wartezeit zwischen den Checkpoints, bevor Sie fortfahren.
- Unter bestimmten Umständen ist es möglich, einen Checkpoint zu überspringen. Angenommen, Ihre Auto-Scaling-Gruppe hat zwei Instances und Ihre Prüfpunkt-Prozentsätze sind $[10, 40, 100]$. Nachdem die erste Instance ersetzt wurde, berechnet Amazon EC2 Auto Scaling, dass 50 Prozent der Gruppe ersetzt wurden. Da 50 Prozent höher ist als die ersten beiden Prüfpunkte, überspringt es den ersten Prüfpunkt (10) und sendet eine Benachrichtigung für den zweiten Prüfpunkt (40).
- Wenn Sie den Vorgang abbrechen, werden alle weiteren Ersetzungen beendet. Wenn Sie den Vorgang abbrechen oder er vor dem Erreichen des letzten Checkpoints fehlschlägt, werden alle Instances, die bereits ersetzt wurden, nicht auf die vorherige Konfiguration zurückgesetzt.

- Bei einer teilweisen Aktualisierung startet Amazon EC2 Auto Scaling beim erneuten Ausführen des Vorgangs nicht ab dem Zeitpunkt des letzten Checkpoints neu und stoppt nicht, wenn nur die älteren Instances ersetzt werden. Es wird jedoch zuerst ältere Instances ersetzen, bevor es neue Instances ersetzt.
- Der Prozentsatz, der tatsächlich abgeschlossen ist, kann höher sein als der Prozentsatz für diesen Checkpoint, wenn der Prozentsatz des Checkpoints im Verhältnis zur Anzahl der Instanzen in der Gruppe zu niedrig ist. Nehmen wir zum Beispiel an, der Prozentsatz des Checkpoints liegt bei 20 Prozent und die Gruppe hat vier Instances. Wenn Amazon EC2 Auto Scaling eine der vier Instances ersetzt, ist der tatsächliche Prozentsatz, der ersetzt wurde (25 Prozent), höher als der Prozentsatz des Checkpoints (20 Prozent).
- Nachdem ein Checkpoint erreicht wurde, wird der angezeigte Gesamtprozentsatz für abgeschlossen erst aktualisiert, nachdem die Instances den Warmlauf abgeschlossen haben. Ihre Checkpoint-Prozentsätze weisen beispielsweise eine Verzögerung von 15 Minuten und einen fehlerfreien Mindestprozentsatz von 80 Prozent auf. [20, 50] Ihre Auto Scaling Scaling-Gruppe hat 10 Instances und nimmt die folgenden Ersetzungen vor:
 - 0:00: Zwei ältere Instances werden durch neue ersetzt.
 - 0:10: Zwei neue Instances schließen das Aufwärmen ab.
 - 0:25: Zwei ältere Instances werden durch neue ersetzt. (Damit der minimale fehlerfreie Prozentsatz beibehalten wird, werden nur zwei Instances ersetzt.)
 - 0:35: Zwei neue Instances schließen das Aufwärmen ab.
 - 0:35: Eine ältere Instance wird durch eine neue ersetzt.
 - 0:45: Eine neue Instance schließt das Aufwärmen ab.

Bei 0:35 hört der Vorgang auf, neue Instance zu starten. Der abgeschlossene Prozentsatz spiegelt die Anzahl der abgeschlossenen Ersetzungen noch nicht genau wider (50 Prozent), da die neue Instance nicht aufgewärmt ist. Nachdem die neue Instance ihre Aufwärmphase um 0:45 Uhr abgeschlossen hat, wird für den Prozentsatz „Abgeschlossen“ ein Wert von 50 Prozent angezeigt.

- Wenn mehrere Checkpoints aktiviert sind, muss der Checkpoint für die Instance-Aktualisierung auf 100% gesetzt werden, um sicherzustellen, dass alle Instances vollständig ersetzt werden. Wenn Sie den letzten Checkpoint auf einen Prozentsatz von weniger als 100% (z. B. 50%) setzen, ersetzt die Instanzaktualisierung keine Instances mehr, sobald dieser Checkpoint-Prozentsatz erreicht ist, und es wird nicht automatisch zu 100% fortgesetzt.

Aktivieren Sie Checkpoints mithilfe von oder AWS-Managementkonsole oder AWS CLI

Sie können das AWS-Managementkonsole oder verwenden, um Checkpoints mit AWS CLI zu aktivieren.

Aktivieren von Prüfpunkten (Konsole)

Sie können Prüfpunkte aktivieren, bevor Sie eine Instance-Aktualisierung starten, um Instances mit einem inkrementellen oder schrittweisen Ansatz zu ersetzen. Dies bietet zusätzliche Zeit für die Überprüfung.

So starten Sie eine Instance-Aktualisierung, die Checkpoints verwendet

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Unten auf der Seite Auto-Scaling-Gruppen wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Instance refresh (Instance-Aktualisierung) unter Active instance refresh (Aktive Instance-Aktualisierung) die Option Start instance refresh (Instance-Aktualisierung starten) aus.
4. Auf der Seite Start instance refresh (Instance-Aktualisierung starten) geben Sie die Werte für Minimum healthy percentage (Minimaler gesunder Prozentsatz) und Instance warmup (Instance-Aufwärmphase) ein.
5. Aktivieren Sie das Kontrollkästchen Enable checkpoints (Prüfpunkte aktivieren).

Dadurch wird ein Feld angezeigt, in dem Sie den prozentualen Schwellenwert für den ersten Checkpoint definieren können.

6. Für Proceed until ____ % of the group is refreshed (Fortfahren, bis ____ % der Gruppe aktualisiert wurde), geben Sie eine Zahl ein (1-100). Dies legt den Prozentsatz für den ersten Prüfpunkt fest.
7. Um einen weiteren Checkpoint hinzuzufügen, wählen Sie Hinzufügen eines Checkpoints aus und definieren Sie dann den Prozentsatz für den nächsten Checkpoint.
8. Um anzugeben, wie lange Amazon EC2 Auto Scaling wartet, nachdem ein Checkpoint erreicht wurde, aktualisieren Sie die Felder in Zwischen Checkpoints **1 hour** warten. Die Zeiteinheit kann Stunden, Minuten oder Sekunden sein.
9. Wenn Sie mit Ihrer Auswahl für die Instance-Aktualisierung fertig sind, klicken Sie auf Instance-Aktualisierung Starten.

Aktivieren von Prüfpunkten (AWS CLI)

Um eine Instance-Aktualisierung mit aktivierten Checkpoints mithilfe von zu starten AWS CLI, benötigen Sie eine Konfigurationsdatei, die die folgenden Parameter definiert:

- `CheckpointPercentages`: Gibt Schwellenwerte für den Prozentsatz der zu ersetzenden Instances an. Diese Schwellenwerte stellen die Checkpoints zur Verfügung. Wenn der Prozentsatz der ersetzten und aufgewärmten Instances einen der angegebenen Schwellenwerte erreicht, wartet der Vorgang eine bestimmte Dauer. Geben Sie die Wartezeit im `CheckpointDelay` in Sekunden an. Wenn der angegebene Zeitraum abgelaufen ist, wird die Instance-Aktualisierung fortgesetzt, bis sie den nächsten Checkpoint erreicht (falls zutreffend).
- `CheckpointDelay`: Gibt die Dauer in Sekunden an, die nach Erreichen eines Prüfpunkts gewartet wird, bevor fortgefahren wird. Wählen Sie einen Zeitraum, der genügend Zeit für die Durchführung Ihrer Überprüfungen bietet.

Der letzt im `CheckpointPercentages`-Array gezeigte Wert ist der Prozentsatz der Auto-Scaling-Gruppe, der erfolgreich ersetzt werden muss. Der Vorgang wechselt zu `Successful` nachdem dieser Prozentsatz der Gruppe erfolgreich ersetzt wurde und jede Instance als abgeschlossen gilt.

So erstellen Sie mehrere Checkpoints

Verwenden Sie den folgenden [start-instance-refresh](#) Beispielfehl, um mehrere Checkpoints zu erstellen. In diesem Beispiel wird eine Instance-Aktualisierung konfiguriert, die zunächst ein Prozent der Auto-Scaling-Gruppe aktualisiert. Nach zehn Minuten Wartezeit aktualisiert sie dann die nächsten 19 Prozent und wartet weitere zehn Minuten. Schließlich aktualisiert es den Rest der Gruppe, bevor der Vorgang abgeschlossen wird.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von `config.json`:

```
{
  "AutoScalingGroupName": "my-asg",
  "Preferences": {
    "InstanceWarmup": 60,
    "MinHealthyPercentage": 80,
    "CheckpointPercentages": [1,20,100],
    "CheckpointDelay": 600
  }
}
```

```
}
```

So erstellen Sie einen einzelnen Checkpoint

Verwenden Sie den folgenden [start-instance-refresh](#) Beispielfehl, um einen einzelnen Checkpoint zu erstellen. In diesem Beispiel wird eine Instance-Aktualisierung konfiguriert, die zunächst 20 Prozent der Auto-Scaling-Gruppe aktualisiert. Nach zehn Minuten Wartezeit aktualisiert sie den Rest der Gruppe, bevor der Vorgang abgeschlossen wird.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von config.json:

```
{
  "AutoScalingGroupName": "my-asg",
  "Preferences": {
    "InstanceWarmup": 60,
    "MinHealthyPercentage": 80,
    "CheckpointPercentages": [20,100],
    "CheckpointDelay": 600
  }
}
```

So aktualisieren Sie die Auto-Scaling-Gruppe teilweise

Verwenden Sie den folgenden [start-instance-refresh](#) Beispielfehl, um nur einen Teil Ihrer Auto Scaling Scaling-Gruppe zu ersetzen und dann vollständig zu beenden. In diesem Beispiel wird eine Instance-Aktualisierung konfiguriert, die zunächst ein Prozent der Auto-Scaling-Gruppe aktualisiert. Nach zehn Minuten Wartezeit aktualisiert sie die nächsten 19 Prozent, bevor der Vorgang abgeschlossen wird.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Inhalt von config.json:

```
{
  "AutoScalingGroupName": "my-asg",
  "Preferences": {
    "InstanceWarmup": 60,
```

```
"MinHealthyPercentage": 80,  
"CheckpointPercentages": [1,20],  
"CheckpointDelay": 600  
}  
}
```

Auto-Scaling-Instances basierend auf der maximalen Instance-Lebensdauer ersetzen

Die maximale Lebensdauer der Instance gibt die maximale Zeit (in Sekunden) an, die eine Instance in Betrieb sein kann, bevor sie beendet und ersetzt wird. Ein häufiger Anwendungsfall könnte eine Anforderung sein, Instances aufgrund interner Sicherheitsrichtlinien oder externer Compliance-Kontrollen nach einem Zeitplan zu ersetzen.

Sie müssen einen Wert von mindestens 86.400 Sekunden (ein Tag) angeben. Um einen zuvor festgelegten Wert zu löschen, geben Sie den neuen Wert „0“ an. Diese Einstellung gilt für alle aktuellen und zukünftigen Instances in Ihrer Auto-Scaling-Gruppe.

Inhalt

- [Überlegungen](#)
- [Maximale Lebensdauer von Instances festlegen](#)
- [Einschränkungen](#)

Überlegungen

Bei der Verwendung dieser Funktion sollten Sie Folgendes beachten:

- Wenn eine ältere Instance ersetzt und eine neue Instance gestartet wird, verwendet die neue Instance die Startvorlage oder Startkonfiguration, die derzeit der Auto-Scaling-Gruppe zugeordnet ist. Wenn Ihre Startvorlage oder Startkonfiguration die Amazon Machine Image (AMI) -ID einer anderen Version Ihrer Anwendung angibt, wird diese Version Ihrer Anwendung automatisch bereitgestellt.
- Wenn Sie die maximale Instance-Lebensdauer zu niedrig einstellen, können Instances schneller als gewünscht ersetzt werden. Amazon EC2 Auto Scaling ersetzt Instances normalerweise einzeln, mit einer Pause zwischen den Ersetzungen. Wenn die angegebene maximale Instance-Lebensdauer jedoch nicht genügend Zeit bietet, um jede Instance einzeln zu ersetzen, muss

Amazon EC2 Auto Scaling mehr als eine Instance gleichzeitig ersetzen. Es können mehrere Instances gleichzeitig ersetzt werden, bis zu 10 Prozent der aktuellen Kapazität Ihrer Auto-Scaling-Gruppe. Um zu vermeiden, dass zu viele Instances gleichzeitig ersetzt werden, legen Sie entweder eine längere maximale Instance-Lebensdauer fest oder verwenden Sie den Instance-Scale-In-Schutz, um vorübergehend zu verhindern, dass einzelne Instances beendet werden. Weitere Informationen finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).

- Standardmäßig erstellt Amazon EC2 Auto Scaling eine neue Skalierung zum Beenden der fehlerhaften Instance und beendet diese anschließend. Während die Instance beendet wird, startet eine andere Skalierungsaktivität eine neue Instance. Sie können dieses Verhalten so ändern, dass es vor dem Beenden gestartet wird, indem Sie eine Instance-Wartungsrichtlinie verwenden. Weitere Informationen finden Sie unter [Wartungsrichtlinien für Instances](#).

Maximale Lebensdauer von Instances festlegen

Wenn Sie eine Auto-Scaling-Gruppe in der Konsole erstellen, können Sie die Einstellung für die maximale Lebensdauer der Instance nicht festlegen. Nachdem die Gruppe erstellt wurde, können Sie sie jedoch bearbeiten, um die maximale Instance-Lebensdauer festzulegen.

So legen Sie die maximale Instance-Lebensdauer für eine Gruppe fest (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Unten auf der Seite Auto-Scaling-Gruppen wird ein geteilter Bereich geöffnet, in dem Informationen zur Gruppe, die Sie ausgewählt haben, angezeigt werden.

3. Wählen Sie auf der Registerkarte Details die Option Erweiterte Konfigurationen, Bearbeiten.
4. Geben Sie bei Maximale Lebensdauer der Instance die maximale Anzahl von Sekunden ein, die eine Instance in Betrieb sein kann.
5. Wählen Sie Aktualisieren aus.

Auf der Registerkarte Activity (Aktivität) können Sie unter Activity history (Aktivitätsverlauf) während des gesamten Verlaufs die Ersetzung von Instances für die Gruppe anzeigen.

So legen Sie die maximale Instance-Lebensdauer für eine Gruppe fest (AWS CLI)

Sie können den auch verwenden AWS CLI , um die maximale Instanzlebensdauer für neue oder bestehende Auto Scaling Scaling-Gruppen festzulegen.

Verwenden Sie für neue Auto Scaling Scaling-Gruppen den [create-auto-scaling-group](#) Befehl.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Im Folgenden finden Sie eine `config.json`-Beispieldatei, in der eine maximale Instance-Lebensdauer von 2592000 Sekunden (30 Tage) angegeben ist.

```
{
  "AutoScalingGroupName": "my-asg",
  "LaunchTemplate": {
    "LaunchTemplateName": "my-launch-template",
    "Version": "$Default"
  },
  "MinSize": 1,
  "MaxSize": 5,
  "MaxInstanceLifetime": 2592000,
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags": []
}
```

Verwenden Sie für bestehende Auto Scaling Scaling-Gruppen den [update-auto-scaling-group](#) Befehl.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-existing-asg --
max-instance-lifetime 2592000
```

So überprüfen Sie die maximale Instance-Lebensdauer für eine Auto-Scaling-Gruppe

Verwenden Sie den Befehl [describe-auto-scaling-groups](#).

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Einschränkungen

- Die maximale Lebensdauer ist nicht für jede Instance genau garantiert: Es ist nicht garantiert, dass die Instances erst am Ende ihrer maximalen Lebensdauer ersetzt werden. In einigen Situationen muss Amazon EC2 Auto Scaling möglicherweise sofort mit dem Ersetzen von Instances beginnen,

nachdem Sie den Parameter für die maximale Lebensdauer der Instance aktualisiert haben. Der Grund für dieses Verhalten ist, dass nicht alle Instances gleichzeitig ersetzt werden sollen.

- Instance Scale-In Protection ausgezeichnet: Amazon EC2 Auto Scaling bietet Instance-Scale-In-Schutz, mit dem Sie kontrollieren können, welche Instances beendet werden können. Wenn dieser Schutz auf einer Instance aktiviert ist, beendet Amazon EC2 Auto Scaling die Instance nicht, auch wenn sie ihre maximale Instance-Lebensdauer erreicht hat.
- Instances, die vor dem Start beendet wurden: Wenn es nur eine Instance in der Auto-Scaling-Gruppe gibt, kann das Feature der maximalen Instance-Lebensdauer zu einem Ausfall führen, da Amazon EC2 Auto Scaling eine Instance beendet und dann standardmäßig eine neue Instance startet. Um dieses Verhalten so zu ändern, dass der Start vor dem Beenden erfolgt, siehe [Wartungsrichtlinien für Instances](#)

Erhöhen oder verringern Sie die Rechenkapazität Ihrer Anwendung durch Skalierung

Skalierung ist die Möglichkeit, die Rechenkapazität Ihrer Anwendung zu erhöhen oder zu verringern. Die Skalierung beginnt mit einem Ereignis bzw. einer Skalierungsaktion, die eine Auto-Scaling-Gruppe anweist, Amazon EC2-Instances entweder zu starten oder zu beenden.

Amazon EC2 Auto Scaling bietet zur bestmöglichen Erfüllung der Anforderungen Ihrer Anwendungen eine Reihe von Möglichkeiten, mit denen Sie die Skalierung anpassen können. Daher ist es wichtig, dass Sie die Anforderungen Ihrer Anwendung gut kennen. Beachten Sie folgende Überlegungen:

- Welche Rolle sollte Amazon EC2 Auto Scaling in der Architektur Ihrer Anwendung spielen? In der Regel wird das Auto Scaling primär als Möglichkeit zur Vergrößerung bzw. Reduzierung von Kapazität betrachtet, Sie können damit jedoch auch dafür sorgen, dass immer dieselbe Anzahl an Servern verwendet wird.
- Welche Kosteneinschränkungen sind für Sie wichtig? Da Amazon EC2 Auto Scaling EC2-Instances verwendet, zahlen Sie nur für die Ressourcen, die Sie nutzen. Die Kenntnis Ihrer Kosteneinschränkungen hilft Ihnen bei der Entscheidung, wann und in welchem Ausmaß Sie Ihre Anwendungen skalieren möchten.
- Welche Metriken sind für Ihre Anwendung wichtig? Amazon CloudWatch unterstützt eine Reihe verschiedener Metriken, die Sie mit Ihrer Auto Scaling Scaling-Gruppe verwenden können.

Inhalt

- [Wählen Sie Ihre Skalierungsmethode aus](#)
- [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#)
- [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#)
- [Manuelle Skalierung für Amazon EC2 Auto Scaling](#)
- [Geplante Skalierung für Amazon EC2 Auto Scaling](#)
- [Dynamische Skalierung für Amazon EC2 Auto Scaling](#)
- [Prädiktive Skalierung für Amazon EC2 Auto Scaling](#)
- [Steuern welche Auto-Scaling-Instances beim Abskalieren beendet werden](#)
- [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#)

Wählen Sie Ihre Skalierungsmethode aus

Amazon EC2 Auto Scaling bietet Ihnen mehrere Möglichkeiten zur Skalierung Ihrer Auto-Scaling-Gruppe.

Eine feste Anzahl von Instances beibehalten

Standardmäßig hat eine Auto-Scaling-Gruppe keine angehängten Skalierungsrichtlinien oder geplanten Aktionen, sodass sie eine feste Größe beibehält. Sobald Sie Ihre Auto-Scaling-Gruppe erstellt haben, beginnt sie mit dem Start von genügend Instances für die gewünschte Kapazität. Sind der Gruppe keine Skalierungsbedingungen zugewiesen, erhält sie ihre gewünschte Kapazität auch dann aufrecht, wenn eine Instance fehlerhaft wird. Amazon EC2 Auto Scaling überwacht den Zustand jeder einzelnen Instance in Ihrer Auto-Scaling-Gruppe. Wenn festgestellt wird, dass eine Instance fehlerhaft geworden ist, wird sie durch eine neue Instance ersetzt. Eine ausführlichere Beschreibung dieses Prozesses finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Manuelles Skalieren

Die manuelle Skalierung ist die einfachste Möglichkeit zur Skalierung Ihrer Auto-Scaling-Gruppe. Sie können entweder die gewünschte Kapazität der Auto Scaling Scaling-Gruppe aktualisieren oder Instances in der Auto Scaling Scaling-Gruppe beenden. Weitere Informationen finden Sie unter [Manuelle Skalierung für Amazon EC2 Auto Scaling](#).

Skalierung nach Zeitplan

Skalierung nach Zeitplan bedeutet, dass Skalierungsaktionen automatisch in Abhängigkeit von Datum und Uhrzeit ausgeführt werden. Dies ist nützlich, wenn Sie genau wissen, wann die Anzahl der Instances in Ihrer Gruppe vergrößert oder verringert werden müssen, weil der Bedarf nach vorhersehbaren Mustern entsteht. Weitere Informationen finden Sie unter [Geplante Skalierung für Amazon EC2 Auto Scaling](#).

Dynamische Skalierung je nach Bedarf

Die dynamische Skalierung ist eine fortgeschrittenere Skalierung von Ressourcen, die es Ihnen ermöglicht, eine Skalierungsrichtlinie zu definieren, die dynamisch die Größe Ihrer Auto-Scaling-Gruppe an die Bedarfsänderungen angepasst. Nehmen wir beispielsweise an, Sie haben eine Webanwendung, die derzeit auf zwei Instances ausgeführt wird, und Sie möchten, dass die CPU-Auslastung der Auto-Scaling-Gruppe bei etwa 50 Prozent bleibt, wenn die Last der Anwendung sich ändert. Diese Methode eignet sich für die Skalierung bei Verkehrsänderungen, wenn Sie nicht

wissen, wann sich der Verkehr ändern wird. Sie können die Skalierungsrichtlinien so konfigurieren, dass sie für Sie reagieren. Es gibt mehrere Richtlinientypen (oder eine Kombination davon), die Sie verwenden können, um auf Verkehrsänderungen zu reagieren. Weitere Informationen finden Sie unter [Dynamische Skalierung für Amazon EC2 Auto Scaling](#).

Proaktiv skalieren

Sie können auch prädiktive und dynamische Skalierung (proaktiver bzw. reaktiver Ansatz) kombinieren, um Ihre EC2-Kapazität schneller zu skalieren. Verwenden Sie die prädiktive Skalierung, um die Anzahl der EC2-Instances in Ihrer Auto-Scaling-Gruppe vor täglichen und wöchentlichen Mustern in Datenverkehrsflüssen zu erhöhen. Weitere Informationen finden Sie unter [Prädiktive Skalierung für Amazon EC2 Auto Scaling](#).

Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe

Skalierungslimits stellen die gewünschte minimale und maximale Gruppengröße für Ihre Auto-Scaling-Gruppe dar. Sie legen Grenzwerte für die Mindest- und die Höchstgröße separat fest.

Die gewünschte Kapazität der Gruppe kann auf einen Wert innerhalb des Bereichs zwischen der Mindest- und Maximalgröße festgelegt werden. Die gewünschte Kapazität darf die Mindestgröße der Gruppe nicht unterschreiten und die maximale Gruppengröße nicht übersteigen.

- **Desired capacity (Gewünschte Kapazität):** Stellt die Anfangskapazität der Auto-Scaling-Gruppe zum Zeitpunkt der Erstellung dar. Eine Auto-Scaling-Gruppe versucht, die gewünschte Kapazität aufrechtzuerhalten. Zu Beginn wird die Anzahl von Instances gestartet, die für die gewünschte Kapazität angegeben ist. Diese Anzahl wird beibehalten, solange der Auto-Scaling-Gruppe keine Skalierungsrichtlinien oder geplanten Aktionen zugeordnet sind.
- **Minimum capacity (Minimale Kapazität):** Stellt die minimale Gruppengröße dar. Wenn Skalierungsrichtlinien festgelegt sind, können sie die gewünschte Kapazität einer Gruppe nicht unter die Mindestkapazität bringen.
- **Maximum capacity (Maximale Kapazität):** Stellt die maximale Gruppengröße dar. Wenn Skalierungsrichtlinien festgelegt sind, können sie die gewünschte Kapazität einer Gruppe nicht über die Höchstkapazität bringen.

Mindest- und Maximalgröße gelten auch in den folgenden Szenarien:

- Manuelles Skalieren Ihrer Auto-Scaling-Gruppe durch Aktualisieren der gewünschten Kapazität

- Ausführen geplanter Aktionen, welche die gewünschte Kapazität aktualisieren. Wenn eine geplante Aktion ohne Angabe einer neuen Mindest- und Maximalgröße für die Gruppe ausgeführt wird, gilt die aktuelle Mindest- bzw. Maximalgröße der Gruppe.

Eine Auto-Scaling-Gruppe versucht immer, die gewünschte Kapazität aufrechtzuerhalten. Wenn eine Instance unerwartet beendet wird (z. B. aufgrund einer Spot Instance-Unterbrechung, eines Fehlers bei der Zustandsprüfung oder eines menschlichen Eingriffs), startet die Gruppe automatisch eine neue Instance, um die gewünschte Kapazität aufrechtzuerhalten.

So verwalten Sie diese Einstellungen in der Konsole

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich unter Auto Scaling Auto Scaling Groups (Auto Scaling-Gruppe) aus.
3. Aktivieren Sie auf der Seite Auto Scaling Groups (Auto-Scaling-Gruppen) das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Zeigen Sie im unteren Abschnitt auf der Registerkarte Details die aktuellen Einstellungen für die gewünschte Kapazität, die Mindest- und die Höchstkapazität an oder ändern Sie sie. Weitere Informationen finden Sie unter [Ändern der gewünschten Kapazität einer vorhandenen Auto-Scaling-Gruppe](#).

Über dem Bereich Details finden Sie Informationen wie die aktuelle Anzahl von Instances in der Auto-Scaling-Gruppe, die gewünschte Kapazität, die Mindest- und die Höchstkapazität sowie eine Spalte. Wenn die Auto Scaling Scaling-Gruppe Instance-Gewichtungen verwendet, können Sie auch die Anzahl der Kapazitätseinheiten ermitteln, die zur gewünschten Kapazität beigetragen haben.

Um Spalten hinzuzufügen oder aus der Liste zu entfernen, wählen Sie das Einstellungssymbol oben auf der Seite aus. Aktivieren oder deaktivieren Sie anschließend für Auto Scaling groups attributes (Auto-Scaling-Gruppenattribute) die einzelnen Spalten und wählen Sie Confirm (Bestätigen) aus.

So überprüfen Sie die Größe der Auto-Scaling-Gruppe nach dem Vornehmen von Änderungen

In der Spalte Instances wird die Anzahl der aktuell ausgeführten Instances angezeigt. Während eine Instance gestartet oder beendet wird, zeigt die Spalte Status den Status Kapazität aktualisieren an, wie im folgenden Image dargestellt.

<input checked="" type="checkbox"/>	Name	Launch template...	Instances	Status	Desired...	Min	Max
<input checked="" type="checkbox"/>	my-asg	my_template Version Def	0	Updating capacity	1	0	1

Warten Sie einige Minuten, und aktualisieren Sie die Ansicht, um den neuesten Status anzuzeigen. Nach Abschluss einer Skalierungsaktivität wird in der Spalte Instances ein aktualisierter Wert angezeigt.

Die Anzahl von Instances und der Status der aktuell ausgeführten Instances werden auch auf der Registerkarte Instance management (Instance-Verwaltung) unter Instances angezeigt.

Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest

CloudWatch sammelt und aggregiert Nutzungsdaten, wie CPU und Netzwerk-I/O, in Ihren Auto Scaling Scaling-Instances. Sie verwenden diese Metriken, um Skalierungsrichtlinien zu erstellen, welche die Anzahl der Instances in Ihrer Auto-Scaling-Gruppe anpassen, wenn der Wert der ausgewählten Metrik steigt und abnimmt.

Sie können angeben, wie lange eine Instance, nachdem sie den InService Status erreicht hat, wartet, bis sie Nutzungsdaten zu den aggregierten Metriken beiträgt. Diese angegebene Zeit wird als Standard-Instance-Warmup bezeichnet. Dadurch wird verhindert, dass die dynamische Skalierung durch Metriken für einzelne Instances beeinträchtigt wird, die noch keinen Anwendungsdatenverkehr verarbeiten und bei denen möglicherweise vorübergehend eine hohe Auslastung von Rechenressourcen auftritt.

Um die Leistung Ihrer Target-Tracking- und Step-Scaling-Richtlinien zu optimieren, empfehlen wir Ihnen dringend, das standardmäßige Instance-Warmup zu aktivieren und zu konfigurieren. Es ist standardmäßig nicht aktiviert oder konfiguriert.

Wenn Sie das standardmäßige Instance-Warmup aktivieren, sollten Sie bedenken, dass Sie verhindern können, dass Instances auf den Mindestwert für den fehlerfreien Zustand angerechnet werden, wenn Sie Instances durch eine Instanzaktualisierung ersetzen, wenn Ihre Auto Scaling Scaling-Gruppe so eingestellt ist, dass sie vor Abschluss der Initialisierung auf den Mindestwert für intakte Instanzen angerechnet werden.

Inhalt

- [Leistungsaspekte der Skalierung](#)
- [Wählen Sie die Standardzeit für das Aufwärmen der Instanz](#)
- [Aktivieren Sie das Standard-Instance-Warmup für eine Gruppe](#)
- [Überprüfen Sie die standardmäßige Aufwärmzeit der Instanz für eine Gruppe](#)
- [Suchen Sie nach Skalierungsrichtlinien mit einer zuvor festgelegten Aufwärmzeit für Instanzen](#)
- [Löschen Sie die zuvor festgelegte Instance-Vorbereitung für eine Skalierungsrichtlinie](#)

Leistungsaspekte der Skalierung

Für die meisten Anwendungen ist es sinnvoll, eine Standardinstanz-Aufwärmzeit festzulegen, die für alle Funktionen gilt, und nicht unterschiedliche Aufwärmzeiten für verschiedene Funktionen. Wenn Sie beispielsweise keine Standardinstanzaufwärmzeit festlegen, verwendet die Instanzaktualisierungsfunktion die Kulanzzzeit für die Integritätsprüfung als Standard-Aufwärmzeit. Wenn Sie über Richtlinien zur Zielverfolgung und schrittweisen Skalierung verfügen, verwenden diese den für die Standard-Abklingzeit festgelegten Wert als Standard-Aufwärmzeit. Wenn Sie über Richtlinien für vorausschauende Skalierung verfügen, haben diese keine standardmäßige Aufwärmzeit.

Während sich Instances in der Warmlaufphase befinden, werden Ihre dynamischen Skalierungsrichtlinien nur dann skaliert, wenn der Metrikwert von Instances, die sich nicht in der Warmlaufphase befinden, höher ist als der Schwellenwert für die Alarmhöhe der Richtlinie (oder die Zielauslastung einer Skalierungsrichtlinie für die Zielverfolgung). Wenn die Nachfrage sinkt, wird die dynamische Skalierung konservativer, um die Verfügbarkeit Ihrer Anwendung zu schützen. Dadurch wird der Umfang der Aktivitäten für die dynamische Skalierung blockiert, bis die neuen Instances vollständig warmlaufen.

Bei der Skalierung berücksichtigt Amazon EC2 Auto Scaling Instances, die sich gerade aufwärmen, als Teil der Kapazität der Gruppe, wenn entschieden wird, wie viele Instances der Gruppe hinzugefügt werden sollen. Daher führen mehrere Sicherheitslücken, für die eine ähnliche Menge an Kapazität hinzugefügt werden muss, zu einer einzigen Skalierungsaktivität. Es ist beabsichtigt, kontinuierlich zu skalieren, ohne dies übermäßig zu tun.

Wenn das standardmäßige Aufwärmen von Instances nicht aktiviert ist, variiert die Zeit, die eine Instance wartet, bevor sie Metriken an die aktuelle Kapazität sendet CloudWatch und diese auf die aktuelle Kapazität anrechnet, von Instance zu Instance. Es besteht also die Möglichkeit, dass

Ihre Skalierungsrichtlinien im Vergleich zur tatsächlich anfallenden Arbeitslast unvorhersehbar funktionieren.

Stellen Sie sich zum Beispiel eine Anwendung mit einem wiederkehrenden on-and-off Workload-Muster vor. Eine prädiktive Skalierungsrichtlinie wird verwendet, um wiederkehrende Entscheidungen darüber zu treffen, ob die Anzahl der Instances erhöht werden soll. Da es keine standardmäßige Aufwärmzeit für Richtlinien zur vorausschauenden Skalierung gibt, tragen die Instances sofort zu den aggregierten Metriken bei. Wenn diese Instances beim Start eine höhere Ressourcennutzung haben, kann das Hinzufügen von Instances zu einem Anstieg der aggregierten Metriken führen. Abhängig davon, wie lange es dauert, bis sich die Nutzung stabilisiert hat, kann sich dies auf alle dynamischen Skalierungsrichtlinien auswirken, die diese Metriken verwenden. Wird der hohe Alarmschwellenwert einer dynamischen Skalierungsrichtlinie überschritten, nimmt die Größe der Gruppe wieder zu. Während sich die neuen Instances im Aufwärmmodus befinden, wird die Skalierung der Aktivitäten blockiert.

Wählen Sie die Standardzeit für das Aufwärmen der Instanz

Der Schlüssel zur Einstellung der standardmäßigen Instance-Vorbereitung besteht darin, zu bestimmen, wie lange Ihre Instances benötigen, bis die Initialisierung abgeschlossen ist und wie lange der Ressourcenverbrauch benötigt, um sich zu stabilisieren, nachdem sie den InService-Status erreicht haben. Achten Sie bei der Wahl der Instance-Aufwärmzeit auf ein optimales Gleichgewicht zwischen der Erfassung von Nutzungsdaten für legitimen Datenverkehr und der Minimierung der Datenerfassung im Zusammenhang mit temporären Nutzungsspitzen beim Start.

Angenommen, Sie haben eine Auto-Scaling-Gruppe mit einem Elastic-Load-Balancing-Load-Balancer verbunden. Wenn der Start neuer Instances abgeschlossen ist, werden sie beim Load Balancer registriert, bevor sie in den Status InService wechseln. Nachdem die Instances den Status InService erreicht haben, kann der Ressourcenverbrauch immer noch vorübergehende Spitzen erleben und braucht Zeit, um sich zu stabilisieren. Beispielsweise dauert die Stabilisierung des Ressourcenverbrauchs für einen Anwendungsserver, der große Assets herunterladen und zwischenspeichern muss, länger als bei einem leichtgewichtigen Webserver, der keine großen Assets herunterzuladen hat. Die Instance-Vorbereitung bietet die Zeitverzögerung, die für die Stabilisierung des Ressourcenverbrauchs erforderlich ist.

Important

Wenn Sie sich nicht sicher sind, wie viel Zeit Sie für die Aufwärmzeit benötigen, können Sie mit 300 Sekunden beginnen. Verringern oder erhöhen Sie sie dann schrittweise, bis Sie die

beste Skalierungsleistung für Ihre Anwendung erhalten. Möglicherweise müssen Sie dies einige Male tun, um es richtig zu machen. Wenn Sie Skalierungsrichtlinien haben, die über eine eigene Aufwärmzeit (`EstimatedInstanceWarmup`) verfügen, können Sie alternativ diesen Wert für den Start verwenden. Weitere Informationen finden Sie unter [Suchen Sie nach Skalierungsrichtlinien mit einer zuvor festgelegten Aufwärmzeit für Instanzen](#).

Erwägen Sie, Lebenszyklus-Hooks für Anwendungsfälle zu verwenden, in denen Konfigurationsaufgaben oder Skripte beim Start ausgeführt werden sollen. Lebenszyklus-Hooks können verzögern, dass neue Instances in Betrieb genommen werden, bis sie die Initialisierung abgeschlossen haben. Sie sind besonders nützlich, wenn Sie Bootstrapping-Skripte haben, die einige Zeit in Anspruch nehmen. Wenn Sie einen Lebenszyklus-Hook hinzufügen, können Sie den Wert der standardmäßigen Instance-Vorbereitung reduzieren. Weitere Informationen über das Verwenden von Lebenszyklus-Hooks finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).

Aktivieren Sie das Standard-Instance-Warmup für eine Gruppe

Sie können die standardmäßige Instance-Vorbereitung aktivieren, wenn Sie eine Auto-Scaling-Gruppe erstellen. Sie können sie auch für vorhandene Gruppen aktivieren.

Wenn Sie die Standardfunktion zum Aufwärmen von Instanzen aktivieren, müssen Sie für die folgenden Funktionen keine Werte mehr für Aufwärmparameter angeben:

- [Instance-Aktualisierung](#)
- [Zielüberwachung der Skalierung](#)
- [Schrittweise Skalierung](#)

Console

So aktivieren Sie die standardmäßige Instance-Vorbereitung für eine Gruppe (Konsole)

Wenn Sie die Auto-Scaling-Gruppe erstellen, wählen Sie auf der Seite `Configure advanced options` (Erweiterte Optionen konfigurieren) unter `Additional Settings` (Zusätzliche Einstellungen) die Option `Enable default instance warmup` (Standardmäßige Instance-Vorbereitung aktivieren). Wählen Sie die Aufwärmzeit, die Sie für Ihre Anwendung benötigen.

AWS CLI

So aktivieren Sie die standardmäßige Instance-Vorbereitung für eine neue Gruppe (AWS CLI)

Um die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe zu aktivieren, fügen Sie die Option `--default-instance-warmup` hinzu und geben Sie einen Wert in Sekunden von 0 bis 3 600 an. Nachdem dies aktiviert wurde, wird ein Wert von `-1` diese Einstellung ausschalten.

Der folgende [create-auto-scaling-group](#) Befehl erstellt eine Auto Scaling Scaling-Gruppe mit dem Namen `my-asg` und aktiviert das standardmäßige Aufwärmen der Instanz mit einem Wert von `120` Sekunden.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --
default-instance-warmup 120 ...
```

Tip

Wenn dieser Befehl einen Fehler auslöst, stellen Sie sicher, dass Sie die AWS CLI lokale Version auf die neueste Version aktualisiert haben.

Console

So aktivieren Sie die standardmäßige Instance-Vorbereitung für eine vorhandene Gruppe (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die AWS-Region aus, in der Sie Ihre Auto-Scaling-Gruppe erstellt haben.
3. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Wählen Sie auf der Registerkarte Details die Option Erweiterte Konfigurationen, Bearbeiten.
5. Wählen Sie für Default Instance Warmup die Aufwärmzeit aus, die Sie für Ihre Anwendung benötigen.
6. Wählen Sie Aktualisieren aus.

AWS CLI

So aktivieren Sie die standardmäßige Instance-Vorbereitung für eine vorhandene Gruppe (AWS CLI)

Im folgenden Beispiel wird der [update-auto-scaling-group](#) Befehl verwendet, um das standardmäßige Aufwärmen der Instanz mit einem Wert von **120** Sekunden für eine bestehende Auto Scaling Scaling-Gruppe mit dem Namen **my-asg** zu aktivieren.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
default-instance-warmup 120
```

Tip

Wenn dieser Befehl einen Fehler ausgibt, stellen Sie sicher, dass Sie die AWS CLI lokale Version auf die neueste Version aktualisiert haben.

Überprüfen Sie die standardmäßige Aufwärmzeit der Instanz für eine Gruppe

Gehen Sie wie folgt vor, um die standardmäßige Aufwärmzeit einer Instanz für eine Auto Scaling Scaling-Gruppe mithilfe von zu überprüfen. AWS CLI

So überprüfen Sie die standardmäßige Instance-Aufwärmzeit für eine Auto Scaling Scaling-Gruppe

Verwenden Sie den folgenden [describe-auto-scaling-groups](#)-Befehl. **my-asg** Ersetzen Sie durch den Namen Ihrer Auto Scaling Scaling-Gruppe.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn",
      ...
      "DefaultInstanceWarmup": 120
    }
  ]
}
```

```

    }
  ]
}

```

Suchen Sie nach Skalierungsrichtlinien mit einer zuvor festgelegten Aufwärmzeit für Instanzen

Um festzustellen, ob Sie Richtlinien haben, für die eine eigene Aufwärmzeit gilt `EstimatedInstanceWarmup`, führen Sie den folgenden Befehl [describe-policies](#) mit dem aus. `AWS CLI` `my-asg` Ersetzen Sie durch den Namen Ihrer Auto Scaling Scaling-Gruppe.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
--query 'ScalingPolicies[?EstimatedInstanceWarmup!=`null`]'
```

Es folgt eine Beispielausgabe.

```
[
  {
    "AutoScalingGroupName": "my-asg",
    "PolicyName": "cpu50-target-tracking-scaling-policy",
    "PolicyARN": "arn",
    "PolicyType": "TargetTrackingScaling",
    "StepAdjustments": [],
    "EstimatedInstanceWarmup": 120,
    "Alarms": [{
      "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
      "AlarmName": "TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e"
    },
    {
      "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
      "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2"
    }
  ]},
  "TargetTrackingConfiguration": {
    "PredefinedMetricSpecification": {
      "PredefinedMetricType": "ASGAverageCPUUtilization"
    }
  },

```

```
    "TargetValue":50.0,  
    "DisableScaleIn":false  
  },  
  "Enabled":true  
},  
  
... additional policies ...  
  
]
```

Löschen Sie die zuvor festgelegte Instance-Vorbereitung für eine Skalierungsrichtlinie

Nachdem Sie das standardmäßige Aufwärmen der Instanz aktiviert haben, aktualisieren Sie alle Skalierungsrichtlinien, für die noch eine eigene Aufwärmzeit gilt, um den zuvor festgelegten Wert zu löschen. Andernfalls wird die standardmäßige Instance-Vorbereitung überschrieben.

Sie können Skalierungsrichtlinien mithilfe der Konsole AWS CLI, oder aktualisieren. AWS SDKs In diesem Abschnitt werden die Schritte für die Konsole behandelt. Wenn Sie das AWS CLI oder verwenden AWS SDKs, stellen Sie sicher, dass Sie die bestehende Richtlinienkonfiguration beibehalten, aber die `EstimatedInstanceWarmup` Eigenschaft entfernen. Wenn Sie eine bestehende Skalierungsrichtlinie aktualisieren, wird die Richtlinie durch die Richtlinie ersetzt, die Sie beim programmgesteuerten Aufruf angeben. [PutScalingPolicy](#) Die ursprünglichen Werte werden nicht beibehalten.

Um die zuvor festgelegte Instance-Vorbereitung für eine Skalierungsrichtlinie (Konsole) zu löschen

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Automatische Skalierung unter Dynamische Skalierungsrichtlinien die gewünschte Richtlinie aus und klicken Sie dann auf Aktionen, Bearbeiten.
4. Löschen Sie zum Beispiel Instance-Warmup den Instance-Aufwärmwert, um stattdessen den Standardwert für das Aufwärmen der Instance zu verwenden.
5. Wählen Sie Aktualisieren aus.

Manuelle Skalierung für Amazon EC2 Auto Scaling

Sie können die Anzahl der EC2-Instances in Ihrer Auto Scaling Scaling-Gruppe jederzeit manuell anpassen. Dieser Vorgang, bei dem die Anzahl der Instanzen manuell geändert wird, wird als manuelle Skalierung bezeichnet. Die manuelle Skalierung ist eine Alternative zur auto Skalierung, insbesondere wenn Sie einmalige Kapazitätsänderungen vornehmen möchten.

Nachdem Sie Ihre Gruppe manuell skaliert haben, nimmt Amazon EC2 Auto Scaling die normalen Auto Scaling-Aktivitäten auf der Grundlage der von Ihnen definierten Skalierungsrichtlinien und geplanten Aktionen wieder auf. Bei Gruppen, bei denen das standardmäßige Aufwärmen von Instanzen aktiviert ist, durchlaufen alle neuen Instances eine Aufwärmphase, bevor sie zu den Metriken beitragen, die für die auto Skalierung verwendet werden. Diese Aufwärmphase hilft dabei, die Gruppe auf der neuen Kapazität zu stabilisieren. Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

Manchmal möchten Sie möglicherweise Skalierungsrichtlinien und geplante Aktionen vorübergehend deaktivieren, bevor Sie eine Gruppe manuell skalieren. Dadurch wird verhindert, dass Konflikte zwischen manuellen Skalierungsaktionen und automatisierten Skalierungsaktivitäten entstehen. Weitere Informationen finden Sie unter [Schalten Sie Skalierungsaktivitäten aus](#).

Inhalt

- [Ändern der gewünschten Kapazität einer vorhandenen Auto-Scaling-Gruppe](#)
- [Beenden einer Instance in Ihrer Auto-Scaling-Gruppe \(AWS CLI\)](#)

Ändern der gewünschten Kapazität einer vorhandenen Auto-Scaling-Gruppe

Wenn Sie die gewünschte Kapazität Ihrer Auto Scaling-Gruppe ändern, verwaltet Amazon EC2 Auto Scaling den Prozess des Startens und Beendens von Instances, um die neue gewünschte Größe zu erreichen.

Console

Ändern der Größe einer Auto-Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.

2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Am unteren Rand der Seite wird ein geteilter Bereich angezeigt.

3. Wählen Sie auf der Registerkarte Details die Option Gruppendetails, Bearbeiten.
4. Erhöhen oder verringern Sie für Gewünschte Kapazität die gewünschte Kapazität. Um beispielsweise die Größe der Gruppe um eins zu erhöhen, geben Sie ein, wenn der aktuelle Wert lautet 12.

Wenn Ihr neuer Wert für die gewünschte Kapazität größer als die gewünschte Mindestkapazität und die gewünschte Höchstkapazität ist, wird die gewünschte Höchstkapazität automatisch auf den neuen Wert für die gewünschte Kapazität erhöht.

5. Wählen Sie Aktualisieren aus, wenn Sie fertig sind.

Stellen Sie sicher, dass die von Ihnen angegebene Gruppengröße dazu geführt hat, dass dieselbe Anzahl von Instances gestartet wurde. Wenn Sie beispielsweise die Gruppengröße um eins erhöht haben, stellen Sie sicher, dass Ihre Auto Scaling Scaling-Gruppe eine zusätzliche Instance gestartet hat.

Überprüfen Sie wie folgt, ob sich die Größe der Auto-Scaling-Gruppe geändert hat:

1. Auf der Registerkarte Aktivität können Sie im Aktivitätsverlauf den Fortschritt der Aktivitäten anzeigen, die der Auto Scaling Scaling-Gruppe zugeordnet sind. In der Status-Spalte wird der aktuelle Status Ihrer Instance angezeigt. Während die Instance gestartet wird, zeigt die Statusspalte `Not yet in service` an. Nach dem Start der Instance ändert sich der Status in `Successful`. Sie können auch das Aktualisierungssymbol verwenden, um den aktuellen Status Ihrer Instance zu sehen. Weitere Informationen finden Sie unter [Eine Skalierung für eine Auto-Scaling-Gruppe überprüfen](#).
2. Auf der Registerkarte Instanzverwaltung unter Instances können Sie den Status der Instance einsehen. Es dauert einige Zeit, bis die Instance startet.
 - In der Spalte Lifecycle (Lebenszyklus) wird Ihnen der Zustand Ihrer Instance angezeigt. Die Instance befindet sich zunächst im Status `Pending`. Wenn eine Instance für den Empfang von Datenverkehr bereit ist, lautet der Status `InService`.
 - In der Spalte Health Status wird das Ergebnis der Amazon EC2 Auto Scaling Scaling-Zustandsprüfungen für Ihre Instance angezeigt.

AWS CLI

Im folgenden Beispiel wird davon ausgegangen, dass Sie eine Auto-Scaling-Gruppe mit einer minimalen Größe von 1 und einer maximalen Größe von 5 erstellt haben. Also verfügt die Gruppe derzeit über eine laufende Instance.

Ändern der Größe einer Auto-Scaling-Gruppe

Verwenden Sie den [set-desired-capacity](#) Befehl, um die Größe Ihrer Auto Scaling Group zu ändern, wie im folgenden Beispiel gezeigt.

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \  
  --desired-capacity 2
```

Wenn Sie die standardmäßige Ruhephase für Ihre Auto-Scaling-Gruppe berücksichtigen möchten, müssen Sie die Option `--honor-cooldown` wie im folgenden Beispiel dargestellt angeben.

Weitere Informationen finden Sie unter [Skalierungsruhephasen für Amazon EC2 Auto Scaling](#).

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \  
  --desired-capacity 2 --honor-cooldown
```

So überprüfen Sie die Größe Ihrer Auto-Scaling-Gruppe

Verwenden Sie den [describe-auto-scaling-groups](#) Befehl, um zu bestätigen, dass sich die Größe Ihrer Auto Scaling Group geändert hat, wie im folgenden Beispiel.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Im Folgenden finden Sie eine Beispielausgabe, die Details zu der Gruppe und den gestarteten Instances enthält.

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "AutoScalingGroupARN": "arn",  
      "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1",  
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
      },  
    },  
  ],  
}
```

```
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 2,
"DefaultCooldown": 300,
"AvailabilityZones": [
  "us-west-2a"
],
"LoadBalancerNames": [],
"TargetGroupARNs": [],
"HealthCheckType": "EC2",
"HealthCheckGracePeriod": 300,
"Instances": [
  {
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1",
      "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-05b4f7d5be44822a6",
    "InstanceType": "t3.micro",
    "HealthStatus": "Healthy",
    "LifecycleState": "Pending"
  },
  {
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1",
      "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-0c20ac468fa3049e8",
    "InstanceType": "t3.micro",
    "HealthStatus": "Healthy",
    "LifecycleState": "InService"
  }
],
"CreatedTime": "2019-03-18T23:30:42.611Z",
"SuspendedProcesses": [],
"VPCZoneIdentifier": "subnet-c87f2be0",
"EnabledMetrics": [],
"Tags": [],
```

```

    "TerminationPolicies": [
      "Default"
    ],
    "NewInstancesProtectedFromScaleIn": false,
    "ServiceLinkedRoleARN": "arn",
    "TrafficSources": []
  }
]
}

```

DesiredCapacity zeigt den neuen Wert. Ihre Auto-Scaling-Gruppe hat eine zusätzliche Instance gestartet.

Beenden einer Instance in Ihrer Auto-Scaling-Gruppe (AWS CLI)

Es kann vorkommen, dass Sie Ihre Auto-Scaling-Gruppe manuell abskalieren möchten, aber eine bestimmte Instance beenden möchten. Sie können Ihre Auto Scaling-Gruppe manuell skalieren, indem Sie den Befehl [terminate-instance-in-auto-scaling-group](#) verwenden und die ID der Instance, die Sie beenden möchten, sowie die `--should-decrement-desired-capacity` Option angeben, wie im folgenden Beispiel gezeigt.

```

aws autoscaling terminate-instance-in-auto-scaling-group \
  --instance-id i-026e4c9f62c3e448c --should-decrement-desired-capacity

```

Im Folgenden finden Sie eine Beispielausgabe, die Details zur Skalierungsaktivität enthält.

```

{
  "Activities": [
    {
      "ActivityId": "b8d62b03-10d8-9df4-7377-e464ab6bd0cb",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-026e4c9f62c3e448c",
      "Cause": "At 2023-09-23T06:39:59Z instance i-026e4c9f62c3e448c was taken out of service in response to a user request, shrinking the capacity from 1 to 0.",
      "StartTime": "2023-09-23T06:39:59.015000+00:00",
      "StatusCode": "InProgress",
      "Progress": 0,
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":\"us-west-2c\"}"
    }
  ]
}

```

}

Diese Option ist in der Konsole nicht verfügbar. Sie können jedoch die Instance-Seite der Amazon EC2 EC2-Konsole verwenden, um eine Instance in Ihrer Auto Scaling Scaling-Gruppe zu beenden. Wenn Sie dies tun, erkennt Amazon EC2 Auto Scaling, dass die Instance nicht mehr läuft, und ersetzt sie automatisch im Rahmen der Zustandsprüfung. Nach dem Beenden der Instance dauert es ein oder zwei Minuten, bis eine neue Instance gestartet wird. Informationen zum Beenden einer Instance finden Sie unter [Terminate an Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Wenn Sie Instances in Ihrer Gruppe beenden und dies zu einer ungleichmäßigen Verteilung auf die Availability Zones führt, gleicht Amazon EC2 Auto Scaling die Gruppe neu aus, um eine gleichmäßige Verteilung wiederherzustellen, sofern Sie den Vorgang nicht unterbrechen. AZRebalance Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).

Geplante Skalierung für Amazon EC2 Auto Scaling

Mit der geplanten Skalierung können Sie eine automatische Skalierung für Ihre Anwendung auf der Grundlage vorhersehbarer Laständerungen einrichten. Sie erstellen geplante Aktionen, die die gewünschte Kapazität Ihrer Gruppe zu bestimmten Zeiten erhöhen oder verringern.

Sie erleben beispielsweise ein regelmäßiges wöchentliches Verkehrsmuster, bei dem die Auslastung unter der Woche zunimmt und gegen Ende der Woche abnimmt. Sie können in Amazon EC2 Auto Scaling einen Skalierungsplan konfigurieren, der diesem Muster entspricht:

- Am Mittwochmorgen erhöht eine geplante Aktion die Kapazität, indem die zuvor festgelegte gewünschte Kapazität der Auto Scaling Scaling-Gruppe erhöht wird.
- Am Freitagabend verringert eine weitere geplante Aktion die Kapazität, indem die zuvor festgelegte gewünschte Kapazität der Auto Scaling Scaling-Gruppe verringert wird.

Mit diesen geplanten Skalierungsaktionen können Sie Kosten und Leistung optimieren. Ihre Anwendung verfügt über ausreichend Kapazität, um die Hauptverkehrsspitzen unter der Woche zu bewältigen, stellt aber zu anderen Zeiten nicht zu viel Kapazität bereit.

Sie können geplante Skalierung und Skalierungsrichtlinien zusammen verwenden, um die Vorteile beider Skalierungsansätze zu nutzen. Nachdem eine geplante Skalierungsaktion ausgeführt wurde, kann die Skalierungsrichtlinie weiterhin Entscheidungen darüber treffen, ob die Kapazität weiter skaliert werden soll. So können Sie sicherstellen, dass Sie über eine ausreichende Kapazität

verfügen, um die Last für Ihre Anwendung zu bewältigen. Während sich Ihre Anwendung an die Nachfrage anpasst, muss die aktuelle Kapazität innerhalb der minimalen und maximalen Kapazität liegen, die durch Ihre geplante Aktion festgelegt wurde.

Inhalt

- [So funktioniert die geplante Skalierung](#)
- [Wiederkehrende Zeitpläne](#)
- [Zeitzone](#)
- [Überlegungen](#)
- [Einschränkungen](#)
- [Eine geplante Aktion erstellen](#)
- [Details zu geplanten Aktionen anzeigen](#)
- [Löschen einer geplanten Aktion](#)

So funktioniert die geplante Skalierung

Um die geplante Skalierung zu verwenden, erstellen Sie geplante Aktionen, die Amazon EC2 Auto Scaling anweisen, Skalierungsaktivitäten zu bestimmten Zeiten durchzuführen. Wenn Sie eine geplante Aktion erstellen, geben Sie die Auto Scaling Scaling-Gruppe an, wann die Skalierungsaktivität stattfinden soll, die neue gewünschte Kapazität und optional eine neue Mindestkapazität und eine neue Höchstkapazität. Sie können geplante Aktionen erstellen, die nur einmal oder nach einem wiederkehrenden Zeitplan skaliert werden.

Zum angegebenen Zeitpunkt skaliert Amazon EC2 Auto Scaling auf der Grundlage der neuen Kapazitätswerte, indem die aktuelle Kapazität mit der angegebenen gewünschten Kapazität verglichen wird.

- Wenn die aktuelle Kapazität unter der angegebenen gewünschten Kapazität liegt, skaliert Amazon EC2 Auto Scaling die angegebene gewünschte Kapazität oder fügt Instances hinzu.
- Wenn die aktuelle Kapazität die angegebene gewünschte Kapazität übersteigt, skaliert Amazon EC2 Auto Scaling Instances auf die angegebene gewünschte Kapazität oder entfernt sie.

Eine geplante Aktion legt die gewünschte, minimale und maximale Kapazität der Gruppe zum angegebenen Datum und zur angegebenen Uhrzeit fest. Sie können eine geplante Aktion jeweils nur für eine dieser Kapazitäten erstellen, z. B. für die gewünschte Kapazität. In einigen Fällen müssen

Sie jedoch die Mindest- und Höchstkapazität angeben, um sicherzustellen, dass die gewünschte Kapazität, die Sie in der Aktion angegeben haben, diese Grenzwerte nicht überschreitet.

Wiederkehrende Zeitpläne

Um mit dem AWS CLI oder einem SDK einen wiederkehrenden Zeitplan zu erstellen, geben Sie einen Cron-Ausdruck und eine Zeitzone an, um zu beschreiben, wann die geplante Aktion wiederholt werden soll. Sie können optional ein Datum und eine Uhrzeit für die Startzeit, die Endzeit oder beides angeben.

Um mit dem einen wiederkehrenden Zeitplan zu erstellen AWS-Managementkonsole, geben Sie das Wiederholungsmuster, die Zeitzone, die Startzeit und optional die Endzeit Ihrer geplanten Aktion an. Alle Wiederholungsmusteroptionen basieren auf Cron-Ausdrücken. Alternativ können Sie Ihren eigenen benutzerdefinierten Cron-Ausdruck schreiben.

Der unterstützte Cron-Ausdruck besteht aus fünf Feldern, getrennt durch Leerzeichen: [Minute] [Stunde] [Tag_des_Monats] [Monat_des_Jahres] [Wochentag]. Beispielsweise konfiguriert der Cron-Ausdruck `30 6 * * 2` eine geplante Aktion, die jeden Dienstag um 6:30 Uhr wiederholt wird. Das Sternchen wird als Platzhalter verwendet, um alle Werte für ein Feld abzugleichen. Weitere Beispiele für Cron-Ausdrücke finden Sie unter <https://crontab.guru/examples.html> Informationen zum Schreiben eigener Cron-Ausdrücke in diesem Format finden Sie unter [Crontab](#).

Wählen Sie Ihre Start- und Endzeiten sorgfältig aus. Beachten Sie Folgendes:

- Wird ein Startzeitpunkt angegeben, führt Amazon EC2 Auto Scaling die Aktion zu diesem Zeitpunkt und dann auf Grundlage der angegebenen Wiederholung aus.
- Wenn Sie eine Endzeit angeben, wird die Aktion nach dieser Zeit nicht mehr wiederholt. Eine geplante Aktion bleibt nicht in Ihrem Konto, nachdem sie ihre Endzeit erreicht hat.
- Wenn eine Wiederholungszeit exakt mit der Endzeit übereinstimmt, führt Amazon EC2 Auto Scaling die geplante Aktion zur Endzeit nicht aus.
- Die Start- und Endzeit müssen in UTC festgelegt werden, wenn Sie das AWS CLI oder ein SDK verwenden.

Zeitzone

Standardmäßig befinden sich die wiederkehrenden Zeitpläne in UTC (Coordinated Universal Time). Sie können die Zeitzone ändern, wenn sie Ihrer örtlichen Zeitzone oder einer Zeitzone in einem

anderen Teil Ihres Netzwerks entsprechen soll. Wenn Sie eine Zeitzone angeben, die Sommerzeit befolgt, wird die Aktion automatisch für Sommerzeit angepasst.

Die gültigen Werte sind die kanonischen Namen für Zeitzonen aus der Zeitzonendatenbank der Internet Assigned Numbers Authority (IANA). Die östliche Zeit der USA wird beispielsweise kanonisch als bezeichnet. `America/New_York` [Weitere Informationen finden Sie unter https://www.iana.org/time-zones](https://www.iana.org/time-zones).

Ortsbezogene Zeitzonen passen sich z. B. `America/New_York` automatisch an die Sommerzeit an. Eine UTC-basierte Zeitzone wie `Etc/UTC` ist eine absolute Zeit und wird nicht der Sommerzeit angepasst.

Sie haben beispielsweise einen wiederkehrenden Zeitplan, dessen Zeitzone `America/New_York` ist. Die erste Skalierungsaktion findet in der `America/New_York`-Zeitzone vor dem Start der Sommerzeit statt. Die nächste Skalierungsaktion findet in der `America/New_York`-Zeitzone nach dem Start der Sommerzeit statt. Die erste Aktion beginnt um 8:00 Uhr UTC-5 Ortszeit, während das zweite Mal um 8:00 Uhr UTC-4 in Ortszeit beginnt.

Wenn Sie eine geplante Aktion mit der erstellen AWS-Managementkonsole und eine Zeitzone angeben, in der die Sommerzeit eingehalten wird, passen sich sowohl der wiederkehrende Zeitplan als auch die Start- und Endzeiten automatisch an die Sommerzeit an.

Überlegungen

Beachten Sie bei der Erstellung einer geplanten Aktion Folgendes:

- Die Reihenfolge der Ausführung geplanter Aktionen wird innerhalb derselben Gruppe, aber nicht gruppenübergreifend, garantiert.
- Eine geplante Aktion wird in der Regel innerhalb von Sekunden ausgeführt. Allerdings kann die Aktion um bis zu zwei Minuten nach der geplanten Startzeit verzögert sein. Da geplante Aktionen innerhalb einer Auto-Scaling-Gruppe in der festgelegten Reihenfolge ausgeführt werden, benötigen Aktionen mit nahe beieinanderliegenden geplanten Startzeiten in der Ausführung mehr Zeit.
- Sie können die geplante Skalierung für eine Auto-Scaling-Gruppe vorübergehend deaktivieren, indem Sie das `ScheduledActions`-Verfahren abbrechen. Dadurch können Sie verhindern, dass geplante Aktionen aktiv sind, ohne sie löschen zu müssen. Sie können die geplante Skalierung dann fortsetzen, wenn Sie sie erneut verwenden möchten. Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).
- Nachdem Sie eine geplante Aktion erstellt haben, können Sie alle Einstellungen mit Ausnahme des Namens aktualisieren.

- Wenn mehrere geplante Aktionen innerhalb derselben Auto Scaling Scaling-Gruppe identische Cron-Ausdrücke haben, wird die Ausführungsreihenfolge willkürlich und undefiniert. Um ein vorhersehbares Verhalten zu gewährleisten, können Sie für jede geplante Aktion eindeutige geplante Startzeiten verwenden.

Einschränkungen

- Die Namen der geplanten Aktionen müssen pro Auto-Scaling-Gruppe eindeutig sein.
- Eine geplante Aktion muss über einen eindeutigen Zeitwert verfügen. Wenn Sie versuchen, eine Aktivität zu einem Zeitpunkt zu planen, zu dem bereits eine andere Skalierung geplant ist, wird der Anruf abgelehnt und gibt einen Fehler zurück, der angibt, dass eine geplante Aktion mit dieser geplanten Startzeit bereits vorhanden ist.
- Sie können maximal 125 geplante Aktionen pro Auto-Scaling-Gruppe erstellen.

Eine geplante Aktion erstellen

Verwenden Sie eine der folgenden Methoden, um eine geplante Aktion für Ihre Auto Scaling Scaling-Gruppe zu erstellen:

Console

Erstellen Sie eine geplante Aktion wie folgt:

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Automatic scaling (Automatische Skalierung) unter Scheduled actions (Geplante Aktionen) die Option Geplante Aktion erstellen (Create scheduled action) aus.
4. Geben Sie einen Namen für die geplante Aktion ein.
5. Für Gewünschte Kapazität, Min., Max. wählen Sie die neue gewünschte Größe der Gruppe und die neuen minimalen und maximalen Größenlimits aus. Die gewünschte Kapazität darf die Mindestgröße der Gruppe nicht unterschreiten und die maximale Gruppengröße nicht übersteigen.

6. Wählen Sie für Recurrence (Wiederholung) eine der verfügbaren Optionen aus.
 - Wenn Sie nach einem wiederkehrenden Zeitplan skalieren möchten, wählen Sie aus, wie oft Amazon EC2 Auto Scaling die geplante Aktion ausführen soll.
 - Wenn Sie eine Option auswählen, die mit Every (Alle) beginnt, wird der Cron-Ausdruck für Sie erstellt.
 - Wenn Sie Cron auswählen, geben Sie einen Cron-Ausdruck ein, der angibt, wann die Aktion ausgeführt werden soll.
 - Wenn Sie nur einmal skalieren möchten, wählen Sie Einmalig aus.

7. Wählen Sie für Zeitzone eine Zeitzone aus. Der Standardwert ist Etc/UTC.

Alle aufgelisteten Zeitzonen stammen aus der IANA-Zeitzonendatenbank. Weitere Informationen finden Sie unter https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

8. Definieren Sie ein Datum und eine Uhrzeit für Bestimmte Startzeit.
 - Wenn Sie einen wiederkehrenden Zeitplan gewählt haben, legt die Startzeit fest, wann die erste geplante Aktion in der wiederkehrenden Reihe ausgeführt wird.
 - Wenn Sie Einmalig als Wiederholung ausgewählt haben, definiert die Startzeit das Datum und die Uhrzeit für die Ausführung der geplanten Aktion.
9. (Optional) Bei wiederkehrenden Zeitplänen können Sie eine Endzeit angeben, indem Sie Festlegen der Endzeit und dann ein Datum und eine Uhrzeit für Beenden bis auswählen.
10. Wählen Sie Erstellen aus. Die Konsole zeigt die geplanten Aktionen der Auto-Scaling-Gruppe an.

AWS CLI

Um eine geplante Aktion zu erstellen, können Sie einen der folgenden Beispielbefehle verwenden. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

Beispiel: Einmalige Skalierung

Verwenden Sie den folgenden Befehl [put-scheduled-update-group-action](#) mit den `--desired-capacity` Optionen `--start-time "YYYY-MM-DDThh:mm:ssZ"` und.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-action \
```

```
--auto-scaling-group-name my-asg --start-time "2021-03-31T08:00:00Z" --desired-capacity 3
```

Beispiel: So planen Sie die Skalierung im Rahmen eines sich wiederholenden Zeitplans

Verwenden Sie den folgenden Befehl [put-scheduled-update-group-action](#) mit den Optionen `--recurrence "cron expression"` und `--desired-capacity`.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \  
--auto-scaling-group-name my-asg --recurrence "0 9 * * *" --desired-capacity 3
```

Standardmäßig führt Amazon EC2 Auto Scaling den angegebenen Wiederholungsplan auf der Grundlage der UTC-Zeitzone aus. Um eine andere Zeitzone anzugeben, schließen Sie die `--time-zone`-Option und den Namen der IANA-Zeitzone an, wie im folgenden Beispiel.

```
--time-zone "America/New_York"
```

Weitere Informationen finden Sie unter https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

Details zu geplanten Aktionen anzeigen

Verwenden Sie eine der folgenden Methoden, um Details zu bevorstehenden geplanten Aktionen für Ihre Auto Scaling Scaling-Gruppe anzuzeigen:

Console

Um Details zu geplanten Aktionen anzuzeigen

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Ihre Auto-Scaling-Gruppe aus.
3. Auf der Registerkarte Automatische Skalierung können Sie sich im Abschnitt Geplante Aktionen über bevorstehende geplante Aktionen informieren.

Beachten Sie, dass die Konsole die Werte für Startzeit und Endzeit in Ihrer Ortszeit anzeigt, wobei der UTC-Offset zum angegebenen Datum und zur angegebenen Uhrzeit gültig ist. Der UTC-Offset

ist die Differenz (in Stunden und Minuten) von Ortszeit zu UTC. Der Wert für Zeitzone zeigt Ihre angeforderte Zeitzone an, z. B. `America/New_York`.

AWS CLI

Verwenden Sie den folgenden [describe-scheduled-actions](#)-Befehl.

```
aws autoscaling describe-scheduled-actions --auto-scaling-group-name my-asg
```

Ist der Befehl erfolgreich, wird eine Ausgabe zurückgegeben, die wie folgt aussehen sollte.

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2020-12-01T00:30:00Z",
      "Time": "2020-12-01T00:30:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4
    }
  ]
}
```

Überprüfen von Skalierungsaktivitäten

Informationen zur Überprüfung der Skalierungsaktivitäten im Zusammenhang mit der geplanten Skalierung finden Sie unter [Eine Skalierung für eine Auto-Scaling-Gruppe überprüfen](#).

Löschen einer geplanten Aktion

Verwenden Sie eine der folgenden Methoden, um eine geplante Aktion zu löschen:

Console

Löschen einer geplanten Aktion

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Ihre Auto-Scaling-Gruppe aus.
3. Wählen Sie auf der Registerkarte Automatic scaling (Automatische Skalierung) unter Scheduled actions (Geplante Aktionen) eine geplante Aktion aus.
4. Wählen Sie Aktionen, Löschen aus.
5. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Ja, löschen.

AWS CLI

Verwenden Sie den folgenden [delete-scheduled-action](#)-Befehl.

```
aws autoscaling delete-scheduled-action --auto-scaling-group-name my-asg \  
--scheduled-action-name my-recurring-action
```

Dynamische Skalierung für Amazon EC2 Auto Scaling

Bei der dynamischen Skalierung wird die Kapazität Ihrer Auto-Scaling-Gruppe skaliert, wenn sich der Datenverkehr ändert.

Amazon EC2 Auto Scaling unterstützt die folgenden Typen von dynamischen Skalierungsrichtlinien:

- Skalierung der Zielverfolgung — Erhöhen und verringern Sie die aktuelle Kapazität der Gruppe auf der Grundlage einer CloudWatch Amazon-Metrik und eines Zielwerts. Das funktioniert ähnlich wie bei einem Thermostat, der die Temperatur in Ihrem Zuhause aufrechterhält: Sie wählen eine Temperatur und der Thermostat erledigt den Rest.
- Step scaling (Schrittweise Skalierung): Erhöht und verringert die aktuelle Kapazität der Gruppe auf der Grundlage einer Reihe von Skalierungsanpassungen, die als Schrittanpassungen bezeichnet werden und je nach Ausmaß der Alarmüberschreitung variieren.
- Simple scaling (Einfache Skalierung): Erhöht und verringert die aktuelle Kapazität der Gruppe auf der Grundlage einer einzelnen Skalierungsanpassung und mit einer Ruhephase zwischen den einzelnen Skalierungsaktivitäten.

Wir empfehlen dringend, dass Sie Skalierungsrichtlinien für die Zielverfolgung verwenden und eine Metrik wählen, die sich umgekehrt proportional zu einer Änderung der Kapazität Ihrer Auto Scaling Scaling-Gruppe ändert. Wenn Sie also die Größe Ihrer Auto Scaling Scaling-Gruppe verdoppeln, sinkt die Metrik um 50 Prozent. Auf diese Weise können die Metrikdaten genau proportionale Skalierungsereignisse auslösen. Enthalten sind Metriken wie die durchschnittliche CPU-Auslastung oder die durchschnittliche Anzahl von Anfragen pro Ziel.

Mit Target Tracking skaliert Ihre Auto Scaling-Gruppe direkt proportional zur tatsächlichen Auslastung Ihrer Anwendung. Das bedeutet, dass eine Zielverfolgungsrichtlinie nicht nur den unmittelbaren Kapazitätsbedarf deckt, indem sie auf Laständerungen reagiert, sondern sich auch an Laständerungen anpassen kann, die im Laufe der Zeit auftreten (beispielsweise aufgrund saisonaler Schwankungen).

Richtlinien zur Zielverfolgung machen es außerdem überflüssig, CloudWatch Alarme und Skalierungsanpassungen manuell zu definieren. Amazon EC2 Auto Scaling verarbeitet dies automatisch auf der Grundlage des von Ihnen festgelegten Ziels.

Inhalt

- [Funktionsweise von dynamischen Skalierungsrichtlinien](#)
- [Mehrere dynamische Skalierungsrichtlinien](#)
- [Skalierungsrichtlinien für die Ziel-Nachverfolgung für Amazon EC2 Auto Scaling](#)
- [Schrittweise und einfache Skalierungsrichtlinien für Amazon EC2 Auto Scaling](#)
- [Skalierungsruhephasen für Amazon EC2 Auto Scaling](#)
- [Skalierungsrichtlinie basierend auf Amazon SQS](#)
- [Eine Skalierung für eine Auto-Scaling-Gruppe überprüfen](#)
- [Eine Skalierungsrichtlinie für eine Auto-Scaling-Gruppe deaktivieren](#)
- [Löschen Sie eine Skalierungsrichtlinie für eine Auto Scaling Scaling-Gruppe](#)
- [Beispiel für Skalierungsrichtlinien für AWS CLI](#)

Funktionsweise von dynamischen Skalierungsrichtlinien

Eine dynamische Skalierungsrichtlinie weist Amazon EC2 Auto Scaling an, eine bestimmte CloudWatch Metrik zu verfolgen, und sie definiert, welche Aktion zu ergreifen ist, wenn der zugehörige CloudWatch Alarm in ALARM ist. Die Metriken, die zum Auslösen des Alarmstatus verwendet werden, sind eine Aggregation von Metriken, die von allen Instances in der Auto-Scaling-Gruppe stammen. (Angenommen, Sie haben eine Auto-Scaling-Gruppe mit zwei Instances, bei

denen eine Instance 60 Prozent CPU und die andere 40 Prozent CPU hat. Im Durchschnitt liegen sie bei 50 Prozent CPU.) Wenn die Richtlinie in Kraft ist, passt Amazon EC2 Auto Scaling die gewünschte Kapazität der Gruppe nach oben oder unten an, wenn die Schwelle eines Alarms überschritten wurde.

Wenn eine dynamische Skalierungsrichtlinie aufgerufen wird und die Kapazitätsberechnung eine Zahl außerhalb des minimalen und maximalen Größenbereichs der Gruppe erzeugt, stellt Amazon EC2 Auto Scaling sicher, dass die neue Kapazität niemals außerhalb der minimalen und maximalen Größengrenzen liegt. Die Kapazität wird auf zwei Arten gemessen: mit denselben Einheiten, die Sie bei der Festlegung der gewünschten Kapazität in Form von Instances ausgewählt haben, oder mit Kapazitätseinheiten (wenn [Instance-Gewichtungen](#) angewendet werden).

- Beispiel 1: Eine Auto-Scaling-Gruppe hat eine maximale Kapazität von 3, eine aktuelle Kapazität von 2 und eine dynamische Skalierungsrichtlinie, die drei Instances hinzufügt. Beim Aufrufen dieser Skalierungsrichtlinie fügt Amazon EC2 Auto Scaling der Gruppe nur eine Instance hinzu, um zu verhindern, dass die Gruppe ihre maximale Größe überschreitet.
- Beispiel 2: Eine Auto-Scaling-Gruppe hat eine Mindestkapazität von 2, eine aktuelle Kapazität von 3 und eine dynamische Skalierungsrichtlinie, die zwei Instances entfernt. Wenn Sie diese Richtlinie aufrufen, entfernt Amazon EC2 Auto Scaling nur eine Instance aus der Gruppe, um zu verhindern, dass die Gruppe ihre Mindestgröße unterschreitet.

Wenn die gewünschte Kapazität die maximale Größengrenze erreicht, stoppt die Skalierung. Wenn der Bedarf sinkt und die aktuelle genutzte Kapazität abnimmt, kann Amazon EC2 Auto Scaling entsprechend aufwärts skalieren.

Die Ausnahme ist, wenn Sie Instance-Gewichte verwenden. In diesem Fall kann Amazon EC2 Auto Scaling über die maximale Größenbeschränkung hinaus skaliert werden, aber nur um das maximale Instance-Gewicht. Die Absicht ist, so nah wie möglich an die neue gewünschte Kapazität zu kommen, aber dennoch die für die Gruppe festgelegten Zuordnungsstrategien einzuhalten. Die Zuweisungsstrategien legen fest, welche Instance-Typen gestartet werden sollen. Die Gewichtungen legen fest, wie viele Kapazitätseinheiten jede Instance auf der Grundlage ihres Instance-Typs zur gewünschten Kapazität der Gruppe beiträgt.

- Beispiel 3: Eine Auto-Scaling-Gruppe hat eine maximale Kapazität von 12, eine aktuelle Kapazität von 10 und eine dynamische Skalierungsrichtlinie, die 5 Kapazitätseinheiten hinzufügt. Den Instance-Typen ist jeweils eine von drei Gewichtungen zugewiesen: 1, 4 oder 6. Wenn Sie die Skalierungsrichtlinie aufrufen, wählt Amazon EC2 Auto Scaling einen Instance-Typ mit einer Gewichtung von 6 basierend auf der Zuordnungsstrategie zum Start. Das Ergebnis dieses Scale-

Out-Ereignisses ist eine Gruppe mit einer gewünschten Kapazität von 12 und einer aktuellen Kapazität von 16.

Mehrere dynamische Skalierungsrichtlinien

In den meisten Fällen reicht eine Skalierungsrichtlinie für die Zielnachverfolgung aus, um Ihre Auto-Scaling-Gruppe für eine automatische Auf- und Abwärtsskalierung zu konfigurieren. Eine Skalierungsrichtlinie für die Ziel-Nachverfolgung ermöglicht es Ihnen, ein gewünschtes Ergebnis auszuwählen und die Auto-Scaling-Gruppe nach Bedarf Instances hinzuzufügen und entfernen zu lassen, um dieses Ergebnis zu erreichen.

Für eine erweiterte Skalierungskonfiguration kann Ihre Auto-Scaling-Gruppe mehr als eine Skalierungsrichtlinie haben. So können Sie beispielsweise eine oder mehrere Skalierungsrichtlinien zur Ziel-Nachverfolgung, eine oder mehrere Richtlinien für schrittweise Skalierung oder beides definieren. Dies bietet eine größere Flexibilität, um mehrere Szenarien abzudecken.

Um zu sehen, wie mehrere dynamische Skalierungsrichtlinien zusammenarbeiten, betrachten Sie eine Anwendung, die eine Auto-Scaling-Gruppe und eine Amazon SQS-Warteschlange verwendet, um Anfragen an eine einzelne EC2-Instance zu senden. Zwei Richtlinien steuern, wann die Auto-Scaling-Gruppe eine horizontale Skalierung nach oben durchführt, um die optimale Leistung der Anwendung sicherzustellen. Eine davon ist eine Zielverfolgungsrichtlinie, die eine benutzerdefinierte Metrik verwendet, um Kapazität basierend auf der Anzahl der SQS-Nachrichten in der Warteschlange hinzuzufügen und zu entfernen. Die andere ist eine schrittweise Skalierungsrichtlinie, die die CloudWatch `CPUUtilization` Amazon-Metrik verwendet, um Kapazität hinzuzufügen, wenn die Instance für einen bestimmten Zeitraum eine Auslastung von 90 Prozent überschreitet.

Wenn mehrere Richtlinien gleichzeitig in Kraft sind, besteht die Möglichkeit, dass jede Richtlinie die Auto-Scaling-Gruppe anweisen könnte, sich gleichzeitig zu vergrößern (oder zu verkleinern). Es ist beispielsweise möglich, dass die `CPUUtilization` Metrik den Schwellenwert des CloudWatch Alarms erreicht und diesen überschreitet, während die benutzerdefinierte SQS-Metrik den Schwellenwert des benutzerdefinierten Metrik-Alarms ansteigt und überschreitet.

In einer solchen Situation wählt Amazon EC2 Auto Scaling die Richtlinie, welche die größte Kapazität für das Aufwärtsskalieren und Abwärtsskalieren bietet. Angenommen, die Richtlinie für `CPUUtilization` startet eine einzelne Instance, während die Richtlinie für die SQS-Warteschlange zwei Instances startet. Sind die Kriterien zur Skalierung nach oben für beide Richtlinien gleichzeitig erfüllt, gibt Amazon EC2 Auto Scaling der SQS-Warteschlange den Vorrang. Daher startet die Auto-Scaling-Gruppe zwei Instances.

Der Ansatz, der Richtlinie mit der größten Kapazität Vorrang einzuräumen, gilt auch dann, wenn die Richtlinien unterschiedliche Kriterien für das Herunterskalieren verwenden. Beispiel: Wenn eine Richtlinie beispielsweise drei Instances beendet, eine andere Richtlinie die Anzahl der Instances um 25 Prozent verringert und die Gruppe zum Zeitpunkt der Abwärtsskalierung über acht Instances verfügt, räumt Amazon EC2 Auto Scaling der Richtlinie Vorrang ein, welche die größte Anzahl von Instances für die Gruppe bereitstellt. Dies führt dazu, dass die Auto-Scaling-Gruppe zwei Instances beendet (25 Prozent von 8 = 2). Damit soll verhindert werden, dass Amazon EC2 Auto Scaling zu viele Instances entfernt.

Sie sollten bei der Verwendung von Zielverfolgungs-Skalierungsrichtlinien mit Schrittskalierungsrichtlinien jedoch vorsichtig sein, da Konflikte zwischen diesen Richtlinien zu unerwünschtem Verhalten führen können. Wenn die Richtlinie zur schrittweisen Skalierung beispielsweise eine Skalierung der Aktivität initiiert, bevor die Zielverfolgungsrichtlinie für die Skalierung bereit ist, wird die Skalierung der Aktivität nicht blockiert. Nach Abschluss der Skalierung der Aktivität könnte die Zielverfolgungsrichtlinie die Gruppe anweisen, erneut zu skalieren.

Skalierungsrichtlinien für die Ziel-Nachverfolgung für Amazon EC2 Auto Scaling

Eine Skalierungsrichtlinie für die Zielverfolgung skaliert automatisch die Kapazität Ihrer Auto Scaling Scaling-Gruppe auf der Grundlage eines Zielmetrikwerts. Sie passt sich automatisch an die individuellen Nutzungsmuster Ihrer individuellen Anwendungen an. Auf diese Weise kann Ihre Anwendung die optimale Leistung und hohe Auslastung Ihrer EC2-Instances aufrechterhalten und so die Kosteneffizienz verbessern, ohne dass manuelles Eingreifen erforderlich ist.

Bei der Ziel-Nachverfolgung wählen Sie eine Metrik und einen Zielwert aus, der die ideale durchschnittliche Auslastung oder den idealen Durchsatz für Ihre Anwendung darstellt. Amazon EC2 Auto Scaling erstellt und verwaltet die CloudWatch Alarmer, die Skalierungsereignisse auslösen, wenn die Metrik vom Ziel abweicht. Dies ähnelt beispielsweise der Art und Weise, wie ein Thermostat eine Zieltemperatur beibehält.

Ein Beispiel: Angenommen, Sie verfügen über eine Webanwendung, die derzeit in zwei Instances ausgeführt wird, und Sie möchten, dass die CPU-Auslastung der Auto-Scaling-Gruppe bei etwa 50 Prozent bleibt, wenn sich die Last der Anwendung ändert. Auf diese Weise erlangen Sie zusätzliche Kapazität für Datenverkehrsspitzen, ohne übermäßig viele Ressourcen im Leerlauf zu verwalten.

Hierzu können Sie eine Skalierungsrichtlinie für die Zielverfolgung erstellen, die eine durchschnittliche CPU-Auslastung von 50 Prozent vorsieht. Dann skaliert Ihre Auto Scaling Scaling-Gruppe die

Kapazität oder erhöht die Kapazität, wenn die CPU 50 Prozent überschreitet, um die erhöhte Last zu bewältigen. Die Kapazität wird erhöht oder verringert, wenn die CPU unter 50 Prozent fällt, um die Kosten in Zeiten geringer Auslastung zu optimieren.

Themen

- [Mehrere Skalierungsrichtlinien für die Zielverfolgung](#)
- [Auswahl von Metriken](#)
- [Definieren des Zielwerts](#)
- [Definieren Sie die Aufwärmzeit der Instanz](#)
- [Überlegungen](#)
- [Erstellen einer Zielverfolgungs-Skalierungsrichtlinie](#)
- [Erstellen Sie eine Richtlinie zur Zielverfolgung mit hochauflösenden Metriken für eine schnellere Reaktion](#)
- [Erstellen Sie mithilfe metrischer Mathematik eine Skalierungsrichtlinie für die Zielverfolgung](#)

Mehrere Skalierungsrichtlinien für die Zielverfolgung

Zur Optimierung der Skalierung können mehrere Skalierungsrichtlinien für die Zielverfolgung miteinander kombiniert werden. Diese müssen allerdings jeweils eine andere Metrik verwenden. Auslastung und Durchsatz können sich beispielsweise gegenseitig beeinflussen. Wenn sich eine dieser Metriken ändert, bedeutet das in der Regel, dass auch andere Metriken betroffen sind. Die Verwendung mehrerer Metriken liefert daher zusätzliche Informationen über die Last, unter der Ihre Auto Scaling Scaling-Gruppe steht. Dies kann Amazon EC2 Auto Scaling dabei helfen, fundiertere Entscheidungen zu treffen, wenn es darum geht, wie viel Kapazität zu Ihrer Gruppe hinzugefügt werden soll.

Die Absicht von Amazon EC2 Auto Scaling besteht darin, der Verfügbarkeit immer Priorität einzuräumen. Es wird die Auto Scaling-Gruppe skaliert, wenn eine der Zielverfolgungsrichtlinien für die Skalierung bereit ist. Die Skalierung erfolgt nur, wenn alle Richtlinien zur Zielverfolgung (wobei die Skalierung in Teilen aktiviert ist) für die Skalierung bereit sind.

Auswahl von Metriken

Sie können Skalierungsrichtlinien zur Zielverfolgung mit vordefinierten oder benutzerdefinierten Metriken erstellen. Vordefinierte Metriken bieten Ihnen einfacheren Zugriff auf die am häufigsten verwendeten Metriken für die Skalierung. Mit benutzerdefinierten Metriken können Sie auf andere

verfügbare CloudWatch Metriken skalieren, einschließlich [hochauflösender Metriken](#), die in kürzeren Intervallen in der Größenordnung von wenigen Sekunden veröffentlicht werden. Sie können Ihre eigenen hochauflösenden Metriken oder Metriken, die von anderen AWS Diensten veröffentlicht werden, veröffentlichen.

Weitere Informationen zum Erstellen von Richtlinien zur Zielverfolgung mithilfe hochauflösender Metriken finden Sie unter [Erstellen Sie eine Richtlinie zur Zielverfolgung mit hochauflösenden Metriken für eine schnellere Reaktion](#).

Target Tracking unterstützt die folgenden vordefinierten Metriken:

- `ASGAverageCPUUtilization` – Durchschnittliche CPU-Nutzung der Auto-Scaling-Gruppe.
- `ASGAverageNetworkIn` – Durchschnittliche Anzahl der von der Auto-Scaling-Gruppe auf allen Netzwerkschnittstellen empfangenen Bytes.
- `ASGAverageNetworkOut` – Durchschnittliche Anzahl der von der Auto-Scaling-Gruppe auf allen Netzwerkschnittstellen gesendeten Bytes.
- `ALBRequestCountPerTarget` – Durchschnittliche Anzahl der Application Load Balancer-Anforderungen pro Ziel für Ihre Auto-Scaling-Gruppe.

Important

Weitere wertvolle Informationen zu den Metriken für CPU-Auslastung, Netzwerk-I/O und Anzahl der Application Load Balancer-Anforderungen pro Ziel finden Sie im Thema [Verfügbare CloudWatch Metriken für Ihre Instances auflisten](#) im Amazon EC2 EC2-Benutzerhandbuch bzw. die [CloudWatch Metriken für Ihren Application Load Balancer](#) im Benutzerhandbuch für Application Load Balancers.

Sie können andere verfügbare CloudWatch Metriken oder Ihre eigenen Metriken auswählen, CloudWatch indem Sie eine benutzerdefinierte Metrik angeben. Ein Beispiel, das eine benutzerdefinierte Metrikspezifikation für eine Skalierungsrichtlinie zur Zielverfolgung mithilfe von spezifiziert AWS CLI, finden Sie unter [Beispiel für Skalierungsrichtlinien für AWS CLI](#).

Berücksichtigen Sie die folgenden Aspekte, wenn Sie eine Metrik auswählen:

- Wir empfehlen, nur Messwerte zu verwenden, die in Intervallen von einer Minute oder weniger verfügbar sind, damit Sie schneller auf Nutzungsänderungen reagieren können. Metriken,

die in kürzeren Intervallen veröffentlicht werden, ermöglichen es der Zielverfolgungsrichtlinie, Änderungen in der Auslastung Ihrer Auto Scaling Group schneller zu erkennen und darauf zu reagieren.

- Wenn Sie vordefinierte Metriken wählen, die von Amazon EC2 veröffentlicht werden, wie z. B. die CPU-Auslastung, empfehlen wir Ihnen, eine detaillierte Überwachung zu aktivieren. Standardmäßig werden alle Amazon EC2-Metriken in Intervallen von fünf Minuten veröffentlicht, sie können jedoch auf ein niedrigeres Intervall von einer Minute konfiguriert werden, indem eine detaillierte Überwachung aktiviert wird. Informationen zur Aktivierung der detaillierten Überwachung finden Sie unter [Überwachung für Auto-Scaling-Instances konfigurieren](#)
- Nicht alle benutzerdefinierten Metriken funktionieren für die Zielverfolgung. Die Metrik muss eine gültige Auslastungsmetrik sein und beschreiben, wie ausgelastet eine Instance ist. Der Wert der Metrik muss sich proportional zur Anzahl der Instances in der Auto-Scaling-Gruppe erhöhen oder verringern. Das muss so sein, damit die Metrikdaten verwendet werden können, um die Anzahl der Instances proportional zu skalieren. Beispielsweise funktioniert die CPU-Auslastung einer Auto-Scaling-Gruppe (d. h. die Amazon EC2-Metrik `CPUUtilization` mit der Metrikdimension `AutoScalingGroupName`), wenn die Last auf der Auto-Scaling-Gruppe auf die Instances verteilt ist.
- Die folgenden Metriken funktionieren nicht für die Ziel-Nachverfolgung:
 - Die Anzahl der Anfragen, die vom Load Balancer empfangen werden, der der Auto-Scaling-Gruppe gegenüber liegt (d. h. die Elastic Load Balancing-Metrik `RequestCount`). Die Anzahl der Anfragen, die vom Load Balancer empfangen werden, ändert sich nicht basierend auf der Auslastung der Auto-Scaling-Gruppe.
 - Load Balancer-Anfragelatenz (d. h. die Elastic Load Balancing-Metrik `Latency`). Die Anfragelatenz kann aufgrund der zunehmenden Nutzung zunehmen, ändert sich aber nicht notwendigerweise proportional.
 - Die CloudWatch Amazon SQS `SQS-ApproximateNumberOfMessagesVisible` Warteschlangenmetrik. Die Anzahl der Nachrichten in einer Warteschlange ändert sich möglicherweise nicht proportional zur Größe der Auto-Scaling-Gruppe, die Nachrichten aus der Warteschlange verarbeitet. Eine benutzerdefinierte Metrik, welche die Anzahl der Nachrichten in der Warteschlange pro EC2-Instance in der Auto-Scaling-Gruppe misst, kann jedoch auch funktionieren. Weitere Informationen finden Sie unter [Skalierungsrichtlinie basierend auf Amazon SQS](#).
- Um die Metrik `ALBRequestCountPerTarget` zu verwenden, müssen Sie den Parameter `ResourceLabel` angeben, um die Load Balancer-Zielgruppe zu identifizieren, die der Metrik zugeordnet ist. Ein Beispiel, das den `ResourceLabel` Parameter für eine Skalierungsrichtlinie

für die Zielverfolgung mithilfe von spezifiziert AWS CLI, finden Sie unter [Beispiel für Skalierungsrichtlinien für AWS CLI](#).

- Wenn eine Metrik echte Werte von 0 ausgibt CloudWatch (z. B. `ALBRequestCountPerTarget`), kann eine Auto Scaling Scaling-Gruppe auf 0 skalieren, wenn über einen längeren Zeitraum kein Datenverkehr zu Ihrer Anwendung erfolgt. Damit Ihre Auto-Scaling-Gruppe auf 0 abskaliert werden kann, wenn keine Anfragen an sie weitergeleitet werden, muss die Mindestkapazität der Gruppe auf 0 festgelegt sein.
- Anstatt neue Metriken zur Verwendung in Ihrer Skalierungsrichtlinie zu veröffentlichen, können Sie mit metrischer Mathematik bestehende Metriken kombinieren. Weitere Informationen finden Sie unter [Erstellen Sie mithilfe metrischer Mathematik eine Skalierungsrichtlinie für die Zielverfolgung](#).

Definieren des Zielwerts

Wenn Sie eine Skalierungsrichtlinie für die Zielverfolgung erstellen, müssen Sie einen Zielwert angeben. Der Zielwert stellt die optimale durchschnittliche Auslastung oder den idealen durchschnittlichen Durchsatz für die Auto-Scaling-Gruppe dar. Für eine kosteneffiziente Ressourcennutzung sollte der Zielwert auf einen möglichst hohen Wert mit einem angemessenen Puffer für unerwartete Datenverkehrserhöhungen festgelegt werden. Wenn Ihre Anwendung optimal für einen normalen Datenverkehrsfluss aufskaliert wird, sollte der tatsächliche Metrikwert dem Zielwert entsprechen oder knapp darunter liegen.

Wenn eine Skalierungsrichtlinie auf dem Durchsatz basiert, z. B. der Anzahl der Anfragen pro Ziel für einen Application Load Balancer, dem Netzwerk-E/A oder anderen Zählmetriken, stellt der Zielwert den optimalen durchschnittlichen Durchsatz einer einzelnen Instance für einen Zeitraum von einer Minute dar.

Definieren Sie die Aufwärmzeit der Instanz

Sie können optional angeben, wie viele Sekunden die Vorbereitung einer neu gestarteten Instance dauert. Bis die angegebene Aufwärmphase abgelaufen ist, wird eine Instance nicht in den aggregierten Metriken der EC2-Instance der Auto-Scaling-Gruppe berücksichtigt.

Während sich die Instances in der Aufwärmphase befinden, werden Ihre Skalierungsrichtlinien nur dann skaliert, wenn der Metrikwert von Instances, die sich nicht in der Warmlaufphase befinden, größer ist als die Zielauslastung der Policy.

Wenn die Gruppe erneut skaliert wird, werden die Instances, die noch vorbereitet werden, als Teil der gewünschten Kapazität für die nächste Aufskalieraktivität gezählt. Der Zweck ist eine kontinuierliche (jedoch nicht exzessive) Erweiterung.

Während der Scale-Out-Aktivität werden alle Skalierungsaktivitäten, die durch Skalierungsrichtlinien initiiert wurden, blockiert, bis die Instances vollständig warmlaufen. Wenn die Instances mit dem Warmlaufen fertig sind und eine Skalierung eintritt, werden alle Instances, die gerade beendet werden, bei der Berechnung der neuen gewünschten Kapazität auf die aktuelle Kapazität der Gruppe angerechnet. Deshalb entfernen wir nicht mehr Instances aus der Auto-Scaling-Gruppe als nötig.

Standardwert

Wenn kein Wert festgelegt ist, verwendet die Skalierungsrichtlinie den Standardwert, bei dem es sich um den Wert für das für die Gruppe definierte [Standardinstanz-Warmup](#) handelt. Wenn das Standard-Aufwärmen der Instanz Null ist, wird auf den Wert der [Standard-Abklingzeit](#) zurückgegriffen. Wir empfehlen, den Standard-Instance-Warmup zu verwenden, um die Aktualisierung aller Skalierungsrichtlinien zu vereinfachen, wenn sich die Aufwärmzeit ändert.

Überlegungen

Bei der Arbeit mit Skalierungsrichtlinien für die Zielverfolgung ist Folgendes zu beachten:

- Erstellen, bearbeiten oder löschen Sie keine CloudWatch Alarmer, die mit einer Skalierungsrichtlinie für die Zielverfolgung verwendet werden. Amazon EC2 Auto Scaling erstellt und verwaltet die CloudWatch Alarmer, die Ihren Ziel-Tracking-Skalierungsrichtlinien zugeordnet sind, und kann sie bei Bedarf bearbeiten, ersetzen oder löschen, um das Skalierungserlebnis für Ihre Anwendungen und deren sich ändernde Nutzungsmuster anzupassen.
- Eine Skalierungsrichtlinie für die Zielverfolgung priorisiert die Verfügbarkeit bei Datenverkehrsschwankungen durch langsames Abskalieren bei nachlassendem Datenverkehr. Wenn Sie mehr Kontrolle wünschen, ist eine Richtlinie zur schrittweisen Skalierung möglicherweise die bessere Option. Sie können den Scale-In-Teil einer Richtlinie zur Zielverfolgung vorübergehend deaktivieren. Dies trägt dazu bei, eine Mindestanzahl von Instanzen für erfolgreiche Bereitstellungen aufrechtzuerhalten.
- Wenn der Metrik Datenpunkte fehlen, führt dies dazu, dass der CloudWatch Alarmstatus auf `INSUFFICIENT_DATA` geändert wird. Amazon EC2 Auto Scaling kann Ihre Gruppe dann erst wieder skalieren, wenn neue Datenpunkte gefunden wurden.

- Wenn die Metrik konstruktionsbedingt nur spärlich gemeldet wird, kann metrische Mathematik hilfreich sein. Um beispielsweise die neuesten Werte zu verwenden, verwenden Sie die Funktion `FILL(m1, REPEAT)`, wobei `m1` die Metrik ist.
- Möglicherweise werden Lücken zwischen den Datenpunkten für den Zielwert und die aktuelle Metrik angezeigt. Der Grund hierfür ist, dass wir konservativ agieren, indem beim Ermitteln der hinzuzufügenden oder zu entfernenden Instances Auf- oder Abrundungen vorgenommen werden. Dies hindert uns daran, eine unzureichende Anzahl von Instances hinzuzufügen oder zu viele Instances zu entfernen. Bei kleineren Auto-Scaling-Gruppen mit weniger Instances scheint die Auslastung der Gruppe jedoch weit vom Zielwert entfernt zu sein.

Nehmen wir beispielsweise an, Sie legen einen Zielwert von 50 Prozent für die CPU-Auslastung fest und Ihre Auto Scaling Scaling-Gruppe überschreitet dann das Ziel. Wir könnten bestimmen, dass durch das Hinzufügen von 1,5 Instances die CPU-Auslastung auf beinahe 50 Prozent sinkt. Da es nicht möglich ist, 1,5 Instances hinzuzufügen, runden wir diesen Wert auf und fügen zwei Instances hinzu. Dadurch wird die CPU-Auslastung möglicherweise auf einen Wert unter 50 Prozent verringert, es wird jedoch sichergestellt, dass Ihre Anwendung über genügend Ressourcen verfügt, um dies zu unterstützen. Wenn wir feststellen, dass das Entfernen von 0,5 Instances Ihre CPU-Auslastung auf über 50 Prozent erhöht, entscheiden wir uns ebenfalls dafür, erst dann zu skalieren, wenn die Metrik so niedrig ist, dass wir glauben, dass die Skalierung keine Schwankungen verursacht.

Bei größeren Auto-Scaling-Gruppen mit mehr Instances wird die Auslastung auf eine größere Anzahl von Instances verteilt, wobei durch das Hinzufügen oder Entfernen von Instances eine kleinere Lücke zwischen dem Zielwert und den tatsächlichen metrischen Datenpunkten entsteht.

- Eine Skalierungsrichtlinie für die Ziel-Nachverfolgung geht davon aus, dass Ihre Auto-Scaling-Gruppe aufskaliert werden soll, wenn die angegebene Metrik über dem Zielwert liegt. Sie können keine Skalierungsrichtlinie für die Ziel-Nachverfolgung verwenden, um Ihre Auto-Scaling-Gruppe zu aufzuskalieren, wenn die angegebene Metrik unter dem Zielwert liegt.

Erstellen einer Zielverfolgungs-Skalierungsrichtlinie

Verwenden Sie eine der folgenden Methoden, um eine Skalierungsrichtlinie für die Zielverfolgung für Ihre Auto Scaling Scaling-Gruppe zu erstellen.

Bevor Sie beginnen, stellen Sie sicher, dass Ihre bevorzugte Metrik in Intervallen von 1 Minute verfügbar ist (im Vergleich zum Standardintervall von 5 Minuten für Amazon-EC2-Metriken).

Console

So erstellen Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung für eine neue Auto-Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
3. Wählen Sie in den Schritten 1, 2 und 3 die gewünschten Optionen aus, und fahren Sie mit Schritt 4: Konfigurieren von Gruppengrößen- und Skalierungsrichtlinien fort.
4. Geben Sie unter Skalierung den Bereich an, zwischen dem Sie skalieren möchten, indem Sie die minimale gewünschte Kapazität und maximale gewünschte Kapazität aktualisieren. Mit diesen beiden Einstellungen kann die Auto-Scaling-Gruppe dynamisch skaliert werden. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).
5. Wählen Sie unter Automatische Skalierung Skalierungsrichtlinie für die Zielnachverfolgung aus.
6. Gehen Sie wie folgt vor, um eine Richtlinie zu definieren:
 - a. Geben Sie einen Namen für die Richtlinie an.
 - b. Wählen Sie unter Metriktyp einen Metriktyp aus.

Wenn Sie Anzahl der Application Load Balancer pro Ziel auswählen, wählen Sie anschließend in Zielgruppe eine Zielgruppe aus.
 - c. Geben Sie einen Target value für die Metrik an.
 - d. (Optional) Aktualisieren Sie bei Instance-Warmup den Instance-Warmup-Wert nach Bedarf.
 - e. (Optional) Wählen Sie Disable scale in to create only a scale-out policy (Abwärtsskalierung deaktivieren, um nur eine Richtlinie für die Aufwärtsskalierung zu erstellen). Auf diese Weise können Sie bei Bedarf eine separate Richtlinie für die horizontale Skalierung nach unten erstellen, die einen anderen Typ aufweist.
7. Fahren Sie mit dem Erstellen der Auto-Scaling-Gruppe fort. Ihre Skalierungsrichtlinie wird erstellt, nachdem die Auto-Scaling-Gruppe erstellt wurde.

So erstellen Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung für eine vorhandene Auto-Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Stellen Sie sicher, dass die Skalierungslimits entsprechend festgelegt sind. Wenn die gewünschte Kapazität der Gruppe z. B. bereits erreicht ist, müssen Sie ein neues Maximum angeben, um eine Aufskalierung durchführen zu können. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).
4. Wählen Sie auf der Registerkarte Automatic scaling (Automatische Skalierung) unter Dynamic scaling policies (Dynamische Skalierungsrichtlinien) die Option Create dynamic scaling policy (Richtlinie für die dynamische Skalierung erstellen) aus.
5. Gehen Sie wie folgt vor, um eine Richtlinie zu definieren:
 - a. Für den Richtlinientyp behalten Sie die Standardeinstellung für Zielverfolgungsskalierung bei.
 - b. Geben Sie einen Namen für die Richtlinie an.
 - c. Wählen Sie unter Metriktyp einen Metriktyp aus. Sie können nur einen Metriktyp auswählen. Um mehr als eine Metrik zu verwenden, erstellen Sie mehrere Richtlinien.

Wenn Sie Anzahl der Application Load Balancer pro Ziel auswählen, wählen Sie anschließend in Zielgruppe eine Zielgruppe aus.

- d. Geben Sie einen Target value für die Metrik an.
 - e. (Optional) Aktualisieren Sie bei Instance-Warmup den Instance-Warmup-Wert nach Bedarf.
 - f. (Optional) Wählen Sie Disable scale in to create only a scale-out policy (Abwärtsskalierung deaktivieren, um nur eine Richtlinie für die Aufwärtsskalierung zu erstellen). Auf diese Weise können Sie bei Bedarf eine separate Richtlinie für die horizontale Skalierung nach unten erstellen, die einen anderen Typ aufweist.
6. Wählen Sie Erstellen aus.

AWS CLI

Um eine Skalierungsrichtlinie für die Zielverfolgung zu erstellen, können Sie das folgende Beispiel verwenden, um Ihnen den Einstieg zu erleichtern. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

Note

Weitere Beispiele finden Sie unter [Beispiel für Skalierungsrichtlinien für AWS CLI](#).

So erstellen Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung (AWS CLI)

1. Verwenden Sie den folgenden `cat` Befehl, um einen Zielwert für Ihre Skalierungsrichtlinie und eine vordefinierte Metrikspezifikation in einer JSON-Datei mit dem Namen `config.json` in Ihrem Home-Verzeichnis zu speichern. Im Folgenden finden Sie ein Beispiel für eine Konfiguration zur Zielverfolgung, mit der die durchschnittliche CPU-Auslastung bei 50 Prozent gehalten wird.

```
$ cat ~/config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification":
  {
    "PredefinedMetricType": "ASGAverageCPUUtilization"
  }
}
```

Weitere Informationen finden Sie [PredefinedMetricSpecification](#) in der Amazon EC2 Auto Scaling API-Referenz.

2. Verwenden Sie den `put-scaling-policy`-Befehl zusammen mit der Datei `config.json`, die Sie im vorherigen Schritt erstellt haben, um Ihre Skalierungsrichtlinie zu erstellen:

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

Bei Erfolg gibt dieser Befehl die Namen ARNs und der beiden CloudWatch Alarme zurück, die in Ihrem Namen erstellt wurden.

```
{
  "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/cpu50-target-tracking-
scaling-policy",
  "Alarms": [
    {
      "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
      "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
    },
    {
      "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
      "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }
  ]
}
```

Erstellen Sie eine Richtlinie zur Zielverfolgung mit hochauflösenden Metriken für eine schnellere Reaktion

Target Tracking unterstützt hochauflösende CloudWatch Metriken mit Datenpunkten auf Sekundenebene, die in kürzeren Intervallen als einer Minute veröffentlicht werden. Konfigurieren Sie Target-Tracking-Richtlinien, um die Auslastung anhand hochauflösender CloudWatch Metriken für Anwendungen mit volatilen Nachfragemustern zu überwachen, z. B. Client-APIs, Live-Streaming-Dienste, E-Commerce-Websites und Datenverarbeitung auf Abruf. Um eine höhere Präzision bei der Abstimmung von Kapazität und Nachfrage zu erreichen, nutzt Target Tracking diese detaillierte Überwachung, um den wechselnden Bedarf und die Auslastung Ihrer EC2-Instances schneller zu erkennen und darauf zu reagieren.

Weitere Informationen darüber, wie Sie Ihre Messwerte mit hoher Auflösung veröffentlichen können, finden Sie unter [Veröffentlichen benutzerdefinierter Metriken](#) im CloudWatch Amazon-Benutzerhandbuch. Um auf EC2-Metriken, wie z. B. die CPU-Auslastung mit hoher Auflösung, zuzugreifen und diese zu veröffentlichen, sollten Sie [CloudWatch Agent](#) verwenden.

AWS-Regionen

Die Zielverfolgung mit hochauflösenden Metriken ist in allen AWS-Regionen außer in verfügbar. AWS GovCloud (US) Regions

So funktioniert die Richtlinie zur Zielverfolgung mit hochauflösenden Metriken

Sie erstellen Richtlinien zur Zielverfolgung, indem Sie die Metrik, die Sie verfolgen möchten, und den Zielwert, den Sie für die Metrik beibehalten möchten, definieren. Um mit einer Metrik mit hoher Auflösung zu skalieren, geben Sie den Namen der Metrik an und legen den Messwertzeitraum, in dem das Ziel-Tracking diese Metrik beobachtet, auf einen Wert unter 60 Sekunden fest. Derzeit beträgt das niedrigste unterstützte Intervall 10 Sekunden. Sie können Ihre Metrik in kürzeren Intervallen als diesen veröffentlichen.

Note

Eine Metrikperiode von mehr als 60 wird nicht unterstützt.

Sie können die Zielverfolgung für eine einzelne CloudWatch Metrik konfigurieren oder mehrere CloudWatch Metriken abfragen und mathematische Ausdrücke verwenden, um neue einzelne Zeitreihen auf der Grundlage dieser Metriken zu erstellen. Mit beiden Optionen können Sie den metrischen Zeitraum definieren.

Beispiele

Beispiel 1

Im folgenden Beispiel wird eine Zielverfolgungsrichtlinie erstellt, die auf einer CloudWatch Metrik mit hoher Auflösung basiert. Die Metrik wird mit einer Auflösung von 10 Sekunden veröffentlicht. Durch die Definition des Zeitraums können Sie die Zielverfolgung aktivieren, um diese Metrik mit einer Genauigkeit von 10 Sekunden zu überwachen. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

```
$ cat ~/config.json
```

```
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
    "MetricName": "MyHighResolutionMetric",
    "Namespace": "MyNamespace",
    "Dimensions": [
      {
        "Name": "MyOptionalDimensionName",
        "Value": "MyOptionalMetricDimensionValue"
      }
    ],
    "Statistic": "Average",
    "Unit": "None"
    "Period": "10"
  }
}
```

Beispiel 2

Sie können metrische mathematische Ausdrücke verwenden, um mehrere Metriken zur Skalierung zu einer einzigen Zeitreihe zu kombinieren. Metrische Mathematik ist besonders nützlich, um bestehende Metriken in Durchschnittswerte pro Instanz umzurechnen. Die Konvertierung von Metriken ist wichtig, da bei der Zielverfolgung davon ausgegangen wird, dass die Metrik umgekehrt proportional zur Kapazität der Auto Scaling Scaling-Gruppe ist. Wenn die Kapazität steigt, sollte die Metrik also um fast das gleiche Verhältnis sinken.

Nehmen wir zum Beispiel an, Sie haben eine Kennzahl, die die ausstehenden Jobs darstellt, die von Ihrer Anwendung bearbeitet werden sollen. Sie können metrische Mathematik verwenden, um die ausstehenden Jobs durch die laufende Kapazität Ihrer Auto Scaling Scaling-Gruppe zu dividieren. Auto Scaling veröffentlicht die Kapazitätsmetrik mit einer Granularität von 1 Minute, sodass für Intervalle unter einer Minute kein Wert für diese Metrik vorhanden ist. Wenn Sie für die Skalierung eine höhere Auflösung verwenden möchten, kann dies zu einem Zeitunterschied zwischen der Kapazität und der Metrik für ausstehende Jobs führen. Um diese Diskrepanz zu vermeiden, empfehlen wir, den FILL-Ausdruck zu verwenden, um die fehlenden Werte mit der Kapazitätsnummer zu füllen, die im Zeitstempel der vorherigen Minute aufgezeichnet wurde.

Im folgenden Beispiel wird metrische Mathematik verwendet, um die Metrik für ausstehende Aufträge durch die Kapazität zu dividieren. Für den Zeitraum legen wir beide Messwerte auf 10 Sekunden fest. Da die Metrik in Intervallen von 1 Minute veröffentlicht wird, verwenden wir den FILL-Vorgang für die Kapazitätsmetrik.

Um metrische Mathematik zu verwenden, um mehrere Metriken zu ändern

```

{
  "CustomizedMetricSpecification": {
    "Metrics": [
      {
        "Label": "Pending jobs to be processed",
        "Id": "m1",
        "MetricStat": {
          "Metric": {
            "MetricName": "MyPendingJobsMetric",
            "Namespace": "Custom",
          },
          "Stat": "Sum"
          "Period": 10
        },
        "ReturnData": false
      },
      {
        "Label": "Get the running instance capacity (matching the period to
that of the m1)",
        "Id": "m2",
        "MetricStat": {
          "Metric": {
            "MetricName": "GroupInServiceInstances",
            "Namespace": "AWS/AutoScaling",
            "Dimensions": [
              {
                "Name": "AutoScalingGroupName",
                "Value": "my-asg"
              }
            ]
          },
          "Stat": "Average"
          "Period": 10
        },
        "ReturnData": false
      },
      {
        "Label": "Calculate the pending job per capacity (note the use of the
FILL expression)",
        "Id": "e1",
        "Expression": "m1 / FILL(m2,REPEAT)",
        "ReturnData": true
      }
    ]
  }
}

```

```
    }  
  ]  
},  
  "TargetValue": 100  
}
```

Überlegungen

Beachten Sie Folgendes, wenn Sie Zielverfolgung und hochauflösende Metriken verwenden.

- Um sicherzustellen, dass keine Datenpunkte fehlen, die zu unerwünschten Ergebnissen der automatischen Skalierung führen könnten, muss Ihre CloudWatch Metrik mit derselben oder einer höheren Auflösung als dem von Ihnen angegebenen Zeitraum veröffentlicht werden.
- Definieren Sie den Zielwert als den per-instance-per-minute metrischen Wert, den Sie für Ihre Auto Scaling Scaling-Gruppe beibehalten möchten. Die Festlegung eines geeigneten Zielwerts ist entscheidend, wenn Sie eine Metrik verwenden, deren Wert sich auf der Grundlage des Zeitraums der Metrik multiplizieren kann. Beispielsweise haben alle zählbasierten Messwerte wie die Anzahl der Anfragen oder ausstehenden Jobs, die die SUM-Statistik verwenden, je nach ausgewähltem Zeitraum einen anderen Metrikerwert. Sie sollten dennoch davon ausgehen, dass Sie sich ein Ziel im Vergleich zum Durchschnitt pro Minute setzen.
- Obwohl für die Nutzung von Amazon EC2 Auto Scaling keine zusätzlichen Gebühren anfallen, müssen Sie für Ressourcen wie Amazon EC2-Instances, CloudWatch Metriken und CloudWatch Alarme bezahlen. Die im vorherigen Beispiel erstellten hochauflösenden Alarme haben einen anderen Preis als die Standardalarme. CloudWatch Weitere Informationen zur CloudWatch Preisgestaltung finden Sie unter [CloudWatch Amazon-Preise](#).
- Für Target Tracking müssen die Kennzahlen die durchschnittliche Auslastung Ihrer EC2-Instances pro Instance wiedergeben. Um dies zu erreichen, können Sie [metrische mathematische Operationen](#) als Teil Ihrer Ziel-Tracking-Richtlinienkonfiguration verwenden. Teilen Sie Ihre Metrik durch die Betriebskapazität Ihrer Auto Scaling Scaling-Gruppe. Stellen Sie sicher, dass für jede der Metriken, die Sie zum Erstellen einer einzelnen Zeitreihe verwenden, derselbe Metrikzeitraum definiert ist. Wenn diese Metriken in unterschiedlichen Intervallen veröffentlicht werden, verwenden Sie den FILL-Vorgang für die Metrik mit dem höheren Intervall, um die fehlenden Datenpunkte auszufüllen.

Erstellen Sie mithilfe metrischer Mathematik eine Skalierungsrichtlinie für die Zielverfolgung

Mithilfe metrischer Mathematik können Sie mehrere CloudWatch Metriken abfragen und mathematische Ausdrücke verwenden, um neue Zeitreihen auf der Grundlage dieser Metriken zu erstellen. Sie können die resultierenden Zeitreihen in der CloudWatch Konsole visualisieren und sie zu Dashboards hinzufügen. Weitere Informationen zur metrischen Mathematik finden Sie unter [Verwenden von metrischer Mathematik](#) im CloudWatch Amazon-Benutzerhandbuch.

Für metrische mathematische Ausdrücke gelten folgende Überlegungen:

- Sie können jede verfügbare CloudWatch Metrik abfragen. Jede Metrik ist eine eindeutige Kombination aus Metrikname, Namespace und null oder mehr Dimensionen.
- Sie können einen beliebigen arithmetischen Operator (+ - */^), jede statistische Funktion (wie AVG oder SUM) oder eine andere Funktion verwenden, die diese CloudWatch Funktion unterstützt.
- Sie können sowohl Metriken als auch die Ergebnisse anderer mathematischer Ausdrücke in den Formeln des mathematischen Ausdrucks verwenden.
- Alle Ausdrücke, die in einer metrischen Spezifikation verwendet werden, müssen letztendlich eine einzige Zeitreihe ergeben.
- Sie können überprüfen, ob ein metrischer mathematischer Ausdruck gültig ist, indem Sie die CloudWatch Konsole oder die CloudWatch [GetMetricData](#)API verwenden.

Beispiel: Amazon-SQS-Warteschlangenrückstand pro Instance

Um den Amazon-SQS-Warteschlangenrückstand pro Instance zu erhalten, nehmen Sie die ungefähre Anzahl der Nachrichten, die für den Abruf aus der Warteschlange zur Verfügung stehen, und dividieren diese Zahl durch die laufende Kapazität der Auto-Scaling-Gruppe im InService-Zustand. Weitere Informationen finden Sie unter [Skalierungsrichtlinie basierend auf Amazon SQS](#).

Die Logik für den Ausdruck lautet wie folgt:

sum of (number of messages in the queue)/(number of InService instances)

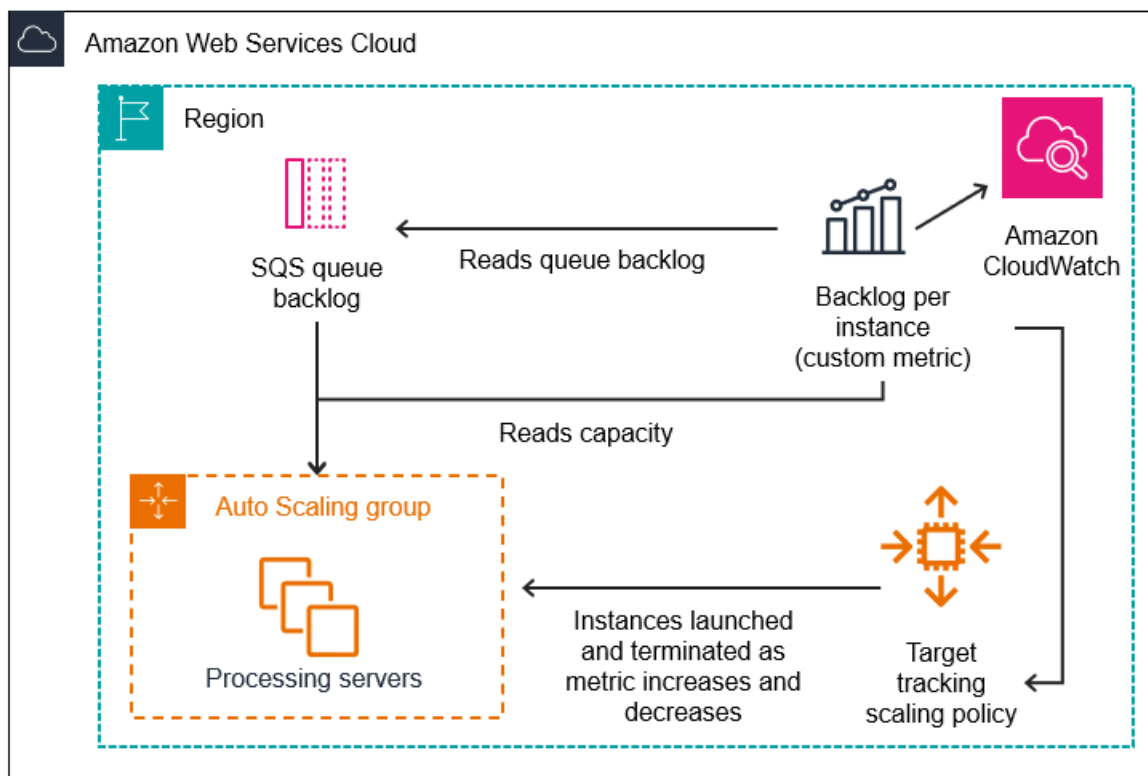
Dann lauten Ihre CloudWatch metrischen Informationen wie folgt.

ID (ID)	CloudWatch Metrik	Statistik	Zeitraum
m1	ApproximateNumberOfMessagesVisible	Summe	1 Minute
m2	GroupInServiceInstances	Durchschnitt	1 Minute

ID und Ausdruck Ihrer Metrikberechnung lauten wie folgt.

ID (ID)	Expression
e1	$(m1)/(m2)$

Das folgende Diagramm veranschaulicht die Architektur dieser Metrik:



So erstellen Sie mithilfe dieser Metrikberechnung eine Skalierungsrichtlinie für die Zielnachverfolgung (AWS CLI)

1. Speichern Sie den metrischen mathematischen Ausdruck als Teil einer benutzerdefinierten Metrikspezifikation in einer JSON-Datei namens `config.json`.

Das folgende Beispiel hilft Ihnen bei den ersten Schritten. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

```
{
  "CustomizedMetricSpecification": {
    "Metrics": [
      {
        "Label": "Get the queue size (the number of messages waiting to be
processed)",
        "Id": "m1",
        "MetricStat": {
          "Metric": {
            "MetricName": "ApproximateNumberOfMessagesVisible",
            "Namespace": "AWS/SQS",
            "Dimensions": [
              {
                "Name": "QueueName",
                "Value": "my-queue"
              }
            ]
          },
          "Stat": "Sum"
        },
        "ReturnData": false
      },
      {
        "Label": "Get the group size (the number of InService instances)",
        "Id": "m2",
        "MetricStat": {
          "Metric": {
            "MetricName": "GroupInServiceInstances",
            "Namespace": "AWS/AutoScaling",
            "Dimensions": [
              {
                "Name": "AutoScalingGroupName",
                "Value": "my-asg"
              }
            ]
          }
        }
      }
    ]
  }
}
```

```

        ],
        },
        "Stat": "Average"
    },
    "ReturnData": false
},
{
    "Label": "Calculate the backlog per instance",
    "Id": "e1",
    "Expression": "m1 / m2",
    "ReturnData": true
}
]
},
"TargetValue": 100
}

```

Weitere Informationen finden Sie [TargetTrackingConfiguration](#) in der Amazon EC2 Auto Scaling API-Referenz.

Note

Im Folgenden finden Sie einige zusätzliche Ressourcen, die Ihnen bei der Suche nach Metrikenamen, Namespaces, Dimensionen und Statistiken für Metriken helfen können: CloudWatch

- Informationen zu den verfügbaren Metriken für AWS Services finden Sie im CloudWatch Amazon-Benutzerhandbuch unter [AWS Services, die CloudWatch Metriken veröffentlichen](#).
- Den genauen Metrikenamen, den Namespace und die Dimensionen (falls zutreffend) für eine CloudWatch Metrik mit dem finden Sie unter AWS CLI [list-metrics](#).

2. Um diese Richtlinie zu erstellen, führen Sie den [put-scaling-policy](#) Befehl mit der JSON-Datei als Eingabe aus, wie im folgenden Beispiel gezeigt.

```

aws autoscaling put-scaling-policy --policy-name sqs-backlog-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json

```

Bei Erfolg gibt dieser Befehl den Amazon-Ressourcennamen (ARN) der Richtlinie und den ARNs der beiden in Ihrem Namen erstellten CloudWatch Alarme zurück.

```
{
  "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
aac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/sqs-backlog-target-
tracking-scaling-policy",
  "Alarms": [
    {
      "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
      "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
    },
    {
      "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
      "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }
  ]
}
```

Note

Wenn dieser Befehl einen Fehler auslöst, stellen Sie sicher, dass Sie die AWS CLI lokale Version auf die neueste Version aktualisiert haben.

Schrittweise und einfache Skalierungsrichtlinien für Amazon EC2 Auto Scaling

Schrittweise Skalierung und einfache Skalierungsrichtlinien skalieren die Kapazität Ihrer Auto Scaling Scaling-Gruppe in vordefinierten Schritten auf der Grundlage von CloudWatch Alarmen. Sie können separate Skalierungsrichtlinien definieren, um die Aufskalierung (Erhöhung der Kapazität)

und die Abskalierung (Verringerung der Kapazität) zu handhaben, wenn ein Alarmschwellenwert überschritten wird.

Die Auto Scaling Scaling-Gruppenkapazität wird in Instanzen oder Kapazitätseinheiten gemessen, wenn Sie [Instanzgewichte](#) verwenden. Außerdem gibt es einen Unterschied zwischen der gewünschten Kapazität und der aktuellen Kapazität.

- **Gewünschte Kapazität** — Die Anzahl der Instanzen (oder Kapazitätseinheiten), die Sie in Ihrer Gruppe haben möchten. Die gewünschte Kapazität kann manuell oder automatisch mithilfe von Skalierungsrichtlinien angepasst werden.
- **Aktuelle Kapazität** — Die Anzahl der Instances (oder Kapazitätseinheiten) in Ihrer Gruppe, die ihre Aufwärm- und Abkühlphase hinter sich haben und nun laufen und einsatzbereit sind.

Mit schrittweiser Skalierung und einfacher Skalierung erstellen und verwalten Sie die CloudWatch Alarmer, die den Skalierungsprozess auslösen. Wenn ein Alarm verletzt wird, initiiert Amazon EC2 Auto Scaling die mit diesem Alarm verknüpfte Skalierungsrichtlinie.

Es wird dringend empfohlen, eine Skalierungsrichtlinie für die Ziel-Nachverfolgung zu verwenden, um auf der Grundlage einer Metrik zu skalieren, z. B. der durchschnittlichen CPU-Auslastung oder der durchschnittlichen Anzahl an Anforderungen pro Ziel. Metriken, die sich verringern, wenn die Kapazität zunimmt, und zunehmen, wenn die Kapazität abnimmt, können zur proportionalen Aufwärts- oder Abwärtsskalierung der Anzahl der Instances verwendet werden, welche die Zielverfolgung verwenden. Dadurch wird sichergestellt, dass Amazon EC2 Auto Scaling die Bedarfskurve für Ihre Anwendungen genau einhält. Weitere Informationen finden Sie unter [Skalierungsrichtlinien für die Ziel-Nachverfolgung](#).

Inhalt

- [Funktionsweise von Skalierungsrichtlinien](#)
- [Schrittanpassungen für die Schrittskalierung](#)
- [Skalierungsanpassungstypen](#)
- [Instance-Aufwärmphase](#)
- [Überlegungen](#)
- [Erstellen Sie eine Richtlinie zur schrittweisen Skalierung für die horizontale Skalierung](#)
- [Erstellen Sie eine Richtlinie zur schrittweisen Skalierung für die Skalierung](#)
- [Einfache Skalierungsrichtlinien](#)

Funktionsweise von Skalierungsrichtlinien

Um Step Scaling zu verwenden, erstellen Sie zunächst einen CloudWatch Alarm, der eine Metrik für Ihre Auto Scaling Scaling-Gruppe überwacht. Sie definieren die Metrik, den Schwellenwert und die Anzahl der Bewertungszeiträume, die einen Alarmverstoß bestimmen. Erstellen Sie dann eine Richtlinie zur schrittweisen Skalierung, die definiert, wie Ihre Gruppe skaliert werden soll, wenn der Alarmschwellenwert überschritten wird. Sie können einen Prozentsatz der aktuellen Kapazität Ihrer Auto Scaling Scaling-Gruppe oder Kapazitätseinheiten für den Skalierungsanpassungstyp verwenden. Weitere Informationen finden Sie unter [Skalierungsanpassungstypen](#).

Sie fügen die schrittweisen Anpassungen in der Richtlinie hinzu. Sie können verschiedene schrittweise Anpassungen basierend auf der Größe der Alarmüberschreitung definieren. Beispiel:

- Skalieren Sie um 10 Instanzen, wenn die Alarmmetrik 60 Prozent erreicht
- Skalieren Sie die Anzeige um 30 Instanzen, wenn die Alarmmetrik 75 Prozent erreicht
- Skalieren Sie die Skala um 40 Instanzen, wenn die Alarmmetrik 85 Prozent erreicht

Wenn der Alarmschwellenwert für die angegebene Anzahl von Testzeiträumen überschritten wird, wendet Amazon EC2 Auto Scaling die in der Richtlinie definierten schrittweisen Anpassungen an. Die Anpassungen können bei weiteren Überschreitungen des Alarms fortgesetzt werden, bis der Alarmstatus OK wieder erreicht ist.

Jede Instance hat eine Aufwärmphase, um zu verhindern, dass Skalierungsaktivitäten zu stark auf Änderungen reagieren, die sich über kurze Zeiträume ergeben. Sie können optional die Aufwärmphase für Ihre Skalierungsrichtlinie konfigurieren. Wir empfehlen jedoch, das standardmäßige Aufwärmen der Instanz zu verwenden, um die Aktualisierung aller Skalierungsrichtlinien zu vereinfachen, wenn sich die Aufwärmzeit ändert. Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

Einfache Skalierungsrichtlinien ähneln schrittweisen Skalierungsrichtlinien, außer dass sie auf einer einzigen Skalierungsanpassung basieren und zwischen den einzelnen Skalierungsaktivitäten eine Abklingzeit besteht. Weitere Informationen finden Sie unter [Einfache Skalierungsrichtlinien](#).

Schrittanpassungen für die Schrittskalierung

Wenn Sie eine Richtlinie zur schrittweise Skalierung erstellen, geben Sie eine oder mehrere Stufenanpassungen an, die automatisch die Anzahl der Instances dynamisch basierend auf der Größe der Alarmüberschreitung skalieren. Jede Schrittanpassung gibt Folgendes an:

- Eine Untergrenze für den Metrikwert
- Eine Obergrenze für den Metrikwert
- Den Skalierungswert basierend auf dem Skalierungsanpassungstyp

CloudWatch aggregiert metrische Datenpunkte auf der Grundlage der Statistik für die Metrik, die Ihrem Alarm zugeordnet ist. CloudWatch Wenn der Alarm ausgelöst wird, wird die entsprechende Skalierungsrichtlinie ausgelöst. Amazon EC2 Auto Scaling wendet den Aggregationstyp auf die neuesten metrischen Datenpunkte von an CloudWatch (im Gegensatz zu den metrischen Rohdaten). Dieser aggregierte Metrikwert wird anschließend mit der Ober- und der Untergrenze verglichen, die durch die Schrittanpassungen definiert wurden. Dadurch wird ermittelt, welche Schrittanpassung auszuführen ist.

Sie geben die Ober- und Untergrenzen relativ zum Verletzungsschwellenwert an. Nehmen wir zum Beispiel an, Sie haben einen CloudWatch Alarm ausgelöst und eine Scale-Out-Richtlinie für den Fall festgelegt, dass die Metrik über 50 Prozent liegt. Dann haben Sie einen zweiten Alarm und eine Abskalierungsrichtlinie für den Fall erstellt, dass die Metrik unter 50 Prozent liegt. Sie haben für jede Richtlinie eine Reihe von schrittweisen Anpassungen mit dem Anpassungstyp `PercentChangeInCapacity` (oder Prozent der Gruppe in der Konsole) vorgenommen:

Beispiel: Schrittanpassungen für die Richtlinie zur horizontalen Skalierung nach oben

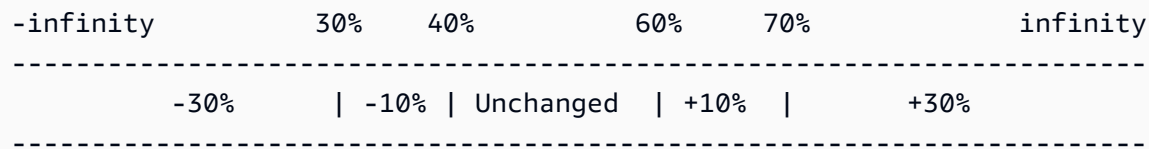
Untergrenze	Obergrenze	Anpassung
0	10	0
10	20	10
20	Null	30

Beispiel: Schrittanpassungen für die Richtlinie zur horizontalen Skalierung nach unten

Untergrenze	Obergrenze	Anpassung
-10	0	0
-20	-10	-10
Null	-20	-30

Dadurch wird die folgende Skalierungskonfiguration erstellt.

Metric value



Nehmen wir nun an, Sie verwenden diese Skalierungskonfiguration für eine Auto Scaling Scaling-Gruppe, die sowohl eine aktuelle als auch eine gewünschte Kapazität von 10 hat. Die folgenden Punkte fassen das Verhalten der Skalierungskonfiguration in Bezug auf die gewünschte und aktuelle Kapazität der Gruppe zusammen:

- Die gewünschte und die aktuelle Kapazität werden aufrechterhalten, solange der aggregierte Metrikerwert größer als 40 und kleiner als 60 ist.
- Steigt der Metrikerwert auf 60, wird die gewünschte Kapazität der Gruppe auf Grundlage der zweiten Schrittanpassung der Richtlinie für die horizontale Skalierung nach oben (Erhöhen um 10 % von 10 Instances) um 1 Instance auf 11 Instances erhöht. Nachdem die neue Instanz ausgeführt wird und die angegebene Aufwärmzeit abgelaufen ist, erhöht sich die aktuelle Kapazität der Gruppe auf 11 Instanzen. Steigt der Metrikerwert auch nach dieser Kapazitätserhöhung auf 70, erhöht sich die gewünschte Kapazität der Gruppe um weitere 3 Instances auf 14 Instances. Dies basiert auf der dritten Schrittanpassung der Richtlinie für die horizontale Skalierung nach oben (Erhöhung um 30 Prozent von 11 Instances, 3,3 Instances, abgerundet auf 3 Instances).
- Fällt der Metrikerwert auf 40 ab, wird die gewünschte Kapazität der Gruppe auf Grundlage der zweiten Schrittanpassung der Richtlinie für die horizontale Skalierung nach unten (Verringern um 10 % von 14 Instances, 1,4 Instances abgerundet auf 1 Instance) um 1 Instance auf 13 Instances reduziert. Wenn der Metrikerwert selbst nach dieser Abnahme der Kapazität auf 30 fällt, sinkt die gewünschte Kapazität der Gruppe um weitere 3 Instances auf 10 Instances. Dies basiert auf der dritten Schrittanpassung der Richtlinie für die horizontale Skalierung nach unten (Abzug um 30 Prozent von 13 Instances, 3,9 Instances, abgerundet auf 3 Instances).

Wenn Sie die Schrittanpassungen für Ihre Skalierungsrichtlinie angeben, beachten Sie Folgendes:

- Wenn Sie die verwenden AWS-Managementkonsole, geben Sie die Ober- und Untergrenzen als absolute Werte an. Wenn Sie das AWS CLI oder ein SDK verwenden, geben Sie die Ober- und Untergrenzen relativ zum Schwellenwert für Sicherheitsverletzungen an.

- Die Bereiche der Schrittanpassungen dürfen sich nicht überschneiden oder Lücken aufweisen.
- Nur eine Schrittanpassung darf über einen Nullwert als Untergrenze verfügen (negative Unendlichkeit). Verfügt eine Schrittanpassung über eine negative Untergrenze, muss eine Schrittanpassung mit einem Nullwert als Untergrenze vorhanden sein.
- Nur eine Schrittanpassung darf über einen Nullwert als Obergrenze verfügen (positive Unendlichkeit). Verfügt eine Schrittanpassung über eine positive Obergrenze, muss eine Schrittanpassung mit einem Nullwert als Obergrenze vorhanden sein.
- Ober- und Untergrenze einer Schrittanpassung können nicht gleichzeitig über einen Nullwert verfügen.
- Liegt der Metrikwert oberhalb des Verletzungsschwellenwerts, wird die Untergrenze eingeschlossen und die Obergrenze ausgeschlossen. Liegt der Metrikwert unterhalb des Verletzungsschwellenwerts, wird die Untergrenze ausgeschlossen und die Obergrenze eingeschlossen.

Skalierungsanpassungstypen

Sie können eine Skalierungsrichtlinie definieren, welche die optimale Skalierungsaktion basierend auf dem von Ihnen gewählten Skalierungsanpassungstyp ausführt. Sie können den Anpassungstyp als Prozentsatz der aktuellen Kapazität Ihrer Auto-Scaling-Gruppe oder in Kapazitätseinheiten angeben. Normalerweise bedeutet eine Kapazitätseinheit eine Instanz, es sei denn, Sie verwenden die Funktion zur Gewichtung von Instanzen.

Amazon EC2 Auto Scaling unterstützt die folgenden Anpassungstypen für die schrittweise und die einfache Skalierung:

- `ChangeInCapacity` – Erhöhen oder Verringern der aktuellen Kapazität der Gruppe um den angegebenen Wert. Ein positiver Wert erhöht die Kapazität, ein negativer Anpassungswert verringert die Kapazität. Beispiel: Wenn die aktuelle Kapazität der Gruppe 3 und die Anpassung 5 beträgt, dann fügen wir bei der Durchführung dieser Richtlinie 5 Kapazitätseinheiten zur Kapazität hinzu, insgesamt also 8 Kapazitätseinheiten.
- `ExactCapacity` – Ändern Sie die aktuelle Kapazität der Gruppe auf den angegebenen Wert. Geben Sie bei diesem Anpassungstyp einen nicht-negativen Wert an. Beispiel: Wenn die aktuelle Kapazität der Gruppe 3 und die Anpassung 5 beträgt, dann ändern wir bei der Durchführung dieser Richtlinie die Kapazität auf 5 Kapazitätseinheiten.
- `PercentChangeInCapacity` – Erhöhen oder Verringern der aktuellen Kapazität der Gruppe um den angegebenen Prozentsatz. Ein positiver Wert erhöht die Kapazität, ein negativer

Anpassungswert verringert die Kapazität. Beispiel: Wenn die aktuelle Kapazität 10 und die Anpassung 10 Prozent beträgt, dann fügen wir bei der Durchführung dieser Richtlinie 1 Kapazitätseinheit zur Kapazität hinzu, insgesamt also 11 Kapazitätseinheiten.

Note

Handelt es sich bei dem resultierenden Wert nicht um eine ganze Zahl, wird wie folgt gerundet:

- Werte größer als 1 werden abgerundet. Beispielsweise wird 12.7 auf 12 gerundet.
- Werte zwischen 0 und 1 werden auf 1 gerundet. Beispielsweise wird .67 auf 1 gerundet.
- Werte zwischen 0 und -1 werden auf -1 gerundet. Beispielsweise wird -.58 auf -1 gerundet.
- Werte kleiner als -1 werden aufgerundet. Beispielsweise wird -6.67 auf -6 gerundet.

Mit `PercentChangeInCapacity` können Sie auch die minimale Anzahl von Instances angeben, die mit dem `MinAdjustmentMagnitude`-Parameter skaliert werden sollen. Angenommen, Sie erstellen eine Richtlinie zum Hinzufügen von 25 % und geben an, dass mindestens 2 Instances hinzugefügt werden sollen. Wenn Sie über eine Auto-Scaling-Gruppe mit 4 Instances verfügen und die Skalierungsrichtlinie umgesetzt wird, ergeben 25 % von 4 Instances 1 Instance. Da Sie aber angegeben haben, dass mindestens 2 Instances hinzugefügt werden sollen, werden 2 Instances hinzugefügt.

Wenn Sie [Instance-Gewichtungen](#) verwenden, ändert sich der Effekt, wenn Sie den `MinAdjustmentMagnitude` Parameter auf einen Wert ungleich Null setzen. Der Wert wird in Kapazitätseinheiten angegeben. Um die Mindestanzahl der zu skalierenden Instances festzulegen, legen Sie diesen Parameter auf einen Wert fest, der mindestens so groß ist wie die größte Instance-Gewichtung.

Wenn Sie Instance-Gewichtungen verwenden, denken Sie daran, dass die aktuelle Kapazität Ihrer Auto Scaling Scaling-Gruppe bei Bedarf die gewünschte Kapazität überschreiten kann. Wenn Ihre absolute Zahl, die zu verringern ist, oder der Betrag, den der Prozentsatz zum Verringern angibt, geringer ist als die Differenz zwischen der aktuellen und der gewünschten Kapazität, wird keine Skalierungsaktion durchgeführt. Sie müssen dieses Verhalten berücksichtigen, wenn Sie sich das Ergebnis einer Skalierungsrichtlinie ansehen, wenn die Schwelle eines Alarms überschritten wird. Angenommen, die gewünschte Kapazität beträgt 30 und die aktuelle Kapazität 32. Wenn beim

Auslösen des Alarms die gewünschte Kapazität mittels Skalierungsrichtlinie um 1 verringert wird, wird keine Skalierungsaktion durchgeführt.

Instance-Aufwärmphase

Sie können für Stufenskalierung optional angeben, wie viele Sekunden die Vorbereitung einer neu gestarteten Instance dauert. Bis die angegebene Aufwärmphase abgelaufen ist, wird eine Instance nicht in den aggregierten Metriken der EC2-Instance der Auto-Scaling-Gruppe berücksichtigt.

Während sich die Instances in der Aufwärmphase befinden, werden Ihre Skalierungsrichtlinien nur dann skaliert, wenn der Metrikwert von Instances, die sich nicht in der Warmlaufphase befinden, über dem Schwellenwert für hohe Alarme der Richtlinie liegt.

Wenn die Gruppe erneut skaliert wird, werden die Instances, die noch vorbereitet werden, als Teil der gewünschten Kapazität für die nächste Aufskalieraktivität gezählt. Daher führen mehrere Alarmüberschreitungen, die in den Bereich derselben Schrittanpassung fallen, zu einer einzigen Skalierung. Der Zweck ist eine kontinuierliche (jedoch nicht exzessive) Erweiterung.

Nehmen wir an, Sie erstellen eine Richtlinie mit zwei Schritten. Im ersten Schritt werden 10 Prozent hinzugefügt, wenn die Metrik 60 erreicht, und im zweiten Schritt werden 30 Prozent hinzugefügt, wenn die Metrik 70 Prozent erreicht. Ihre Auto-Scaling-Gruppe hat eine gewünschte und eine aktuelle Kapazität von 10. Die gewünschte und die aktuelle Kapazität ändern sich nicht, solange der aggregierte Metrikwert kleiner als 60 ist. Nehmen wir an, die Metrik erreicht den Wert 60, sodass 1 Instance hinzugefügt wird (10 Prozent von 10 Instances). Dann erreicht die Metrik den Wert 62, während die neue Instance sich noch in der Aufwärmphase befindet. Die Skalierungsrichtlinie berechnet die neue gewünschte Kapazität auf Grundlage der aktuellen Kapazität, die immer noch 10 beträgt. Allerdings ist die gewünschte Kapazität der Gruppe bereits auf 11 Instances gestiegen, weswegen die Skalierungsrichtlinie die gewünschte Kapazität nicht weiter erhöht. Erreicht die Metrik jedoch den Wert 70, während sich die neue Instance noch in der Aufwärmphase befindet, müssen wir 3 Instances (30 % von 10 Instances) hinzufügen. Die gewünschte Kapazität der Gruppe beträgt jedoch bereits 11, sodass wir für die jetzt gewünschte Kapazität von 13 Instances nur weitere 2 Instances hinzufügen.

Während die Aufskalieraktivität läuft, werden alle durch Skalierungsrichtlinien initiierte Abskalieraktivitäten blockiert, bis die Instances vorbereitet wurden. Wenn die Instances mit dem Aufwärmen fertig sind und ein Abskalierungsereignis eintritt, werden alle Instances, die gerade beendet werden, bei der Berechnung der neuen gewünschten Kapazität auf die aktuelle Kapazität der Gruppe angerechnet. Deshalb entfernen wir nicht mehr Instances aus der Auto-Scaling-Gruppe als nötig. Wenn beispielsweise eine Instance bereits beendet ist und ein Alarm im Bereich der

gleichen Schrittanpassung auftritt, die die gewünschte Kapazität um 1 verringert hat, wird keine Skalierungsmaßnahme ergriffen.

Standardwert

Wenn kein Wert festgelegt ist, verwendet die Skalierungsrichtlinie den Standardwert. Dabei handelt es sich um den Wert für das für die Gruppe definierte [Standardinstanz-Warmup](#). Wenn das Standard-Aufwärmen der Instanz Null ist, wird auf den Wert der [Standard-Abklingzeit](#) zurückgegriffen.

Überlegungen

Bei der Arbeit mit Richtlinien zur schrittweisen und einfachen Skalierung ist Folgendes zu beachten:

- Überlegen Sie, ob Sie die Schrittanpassungen in der Anwendung genau genug vorhersagen können, um die schrittweise Skalierung zu verwenden. Wenn Ihre Skalierungsmetrik die Kapazität des skalierbaren Ziels proportional vergrößert oder verkleinert, raten wir stattdessen zur Verwendung einer Skalierungsrichtlinie für die Ziel-Nachverfolgung. Sie haben weiterhin die Möglichkeit, die Schrittskalierung als zusätzliche Richtlinie für eine erweiterte Konfiguration zu verwenden. Beispiel: Sie können eine striktere Antwort konfigurieren, sobald die Auslastung ein bestimmtes Niveau erreicht.
- Achten Sie darauf, einen angemessenen Abstand zwischen den Schwellenwerten für Scale-Out und Scale-In zu wählen, um ein Flattern zu verhindern. Flattern beschreibt eine Endlosschleife aus Auf- und Abwärtsskalieren. Das heißt, wenn eine Skalierungsaktion durchgeführt wird, würde sich der Metrikwert ändern und eine weitere Skalierungsaktion in der umgekehrten Richtung starten.

Erstellen Sie eine Richtlinie zur schrittweisen Skalierung für die horizontale Skalierung


Verwenden Sie eine der folgenden Methoden, um eine schrittweise Skalierungsrichtlinie für die horizontale Skalierung für Ihre Auto Scaling-Gruppe zu erstellen:

Console

Schritt 1: Erstellen Sie einen CloudWatch Alarm für den hohen Schwellenwert der Metrik

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Ändern Sie, falls erforderlich, die Region. Wählen Sie auf der Navigationsleiste die Region aus, in der sich Ihre Auto-Scaling-Gruppe befindet.
3. Wählen Sie im Navigationsbereich Alarms > All alarms (Alarme > Alle Alarme) und anschließend Create alarm (Alarm erstellen) aus.

4. Wählen Sie **Select metric (Metrik auswählen)** aus.
5. Wählen Sie auf der Registerkarte **Alle Metriken** die Option **EC2 und Nach Auto-Scaling-Gruppe** aus und geben Sie den Namen der Auto-Scaling-Gruppe in das Suchfeld ein. Wählen Sie dann **CPUUtilization** und anschließend **Metrik auswählen** aus. Die Seite **Specify metric and conditions (Metrik und Bedingungen festlegen)** mit einem Diagramm und weiteren Informationen über die Metrik werden angezeigt.
6. Wählen Sie unter **Period (Zeitraum)** den Auswertungszeitraum für den Alarm aus, z. B. 1 Minute. Beim Auswerten des Alarms wird jeder Zeitraum in einem Datenpunkt zusammengefasst.

 **Note**

Ein kürzerer Zeitraum erzeugt eine höhere Alarmempfindlichkeit.

7. Führen Sie unter **Bedingungen** die folgenden Schritte aus:
 - Wählen Sie für **Threshold type (Schwellenwerttyp)** die Option **Static (Statisch)** aus.
 - Geben Sie für **Whenever **CPUUtilization** is an**, ob der Wert der Metrik größer oder größer als oder gleich dem Schwellenwert sein soll, ab dem der Alarm überschritten werden kann. Geben Sie dann unter **than (als)** den Schwellenwert ein, der den Alarm auslösen soll.
8. Führen Sie unter **Zusätzliche Konfiguration** die folgenden Schritte aus:
 - Geben Sie unter **Datenpunkte zum Alarm** die Anzahl der Datenpunkte (Auswertungszeiträume) ein, während denen der Metrikwert die Schwellenbedingungen des Alarms erfüllen muss. So würde es bei zwei aufeinanderfolgenden Zeiträumen von je 5 Minuten z. B. 10 Minuten dauern, den Alarmstatus auszulösen.
 - Wählen Sie für **Fehlende Datenbehandlung** die Option **Fehlende Daten als ungültig behandeln (Überschreitungsschwelle)** aus. Weitere Informationen finden Sie unter [Konfiguration der Behandlung fehlender Daten durch CloudWatch Alarme](#) im **CloudWatch Amazon-Benutzerhandbuch**.
9. Wählen Sie **Weiter** aus.

Die Seite **Configure actions (Konfigurieren von Aktionen)** wird angezeigt.
10. Wählen Sie unter **Notification (Benachrichtigung)** ein Amazon-SNS-Thema aus, das benachrichtigt werden soll, wenn sich der Alarm im Zustand **ALARM**, **OK** oder **INSUFFICIENT_DATA** befindet.

Um zu erreichen, dass der Alarm mehrere Benachrichtigungen für den gleichen Alarmstatus oder für verschiedene Statuswerte sendet, wählen Sie Benachrichtigung hinzufügen.

Damit der Alarm keine Benachrichtigungen sendet, wählen Sie Remove (Entfernen).

11. Sie können die anderen Abschnitte der Seite Configure actions (Konfigurieren von Aktionen) leer lassen. Wenn Sie die anderen Abschnitte leer lassen, wird ein Alarm erstellt, ohne diesen einer Skalierungsrichtlinie zuzuordnen. Sie können den Alarm dann über die Amazon EC2 Auto Scaling-Konsole mit einer Skalierungsrichtlinie verknüpfen.
12. Wählen Sie Weiter aus.
13. Geben Sie einen Namen (beispielsweise Step-Scaling-AlarmHigh-AddCapacity) und optional eine Beschreibung des Alarms ein. Wählen Sie anschließend Next (Weiter) aus.
14. Wählen Sie Alarm erstellen aus.

Gehen Sie wie folgt vor, um dort weiterzumachen, wo Sie nach der Erstellung Ihres CloudWatch Alarms aufgehört haben.

Schritt 2: Erstellen Sie eine Richtlinie zur schrittweisen Skalierung für die horizontale Skalierung

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Stellen Sie sicher, dass die Skalierungslimits entsprechend festgelegt sind. Wenn die gewünschte Kapazität der Gruppe z. B. bereits erreicht ist, müssen Sie ein neues Maximum angeben, um eine Aufskalierung durchführen zu können. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).
4. Wählen Sie auf der Registerkarte Automatic scaling (Automatische Skalierung) unter Dynamic scaling policies (Dynamische Skalierungsrichtlinien) die Option Create dynamic scaling policy (Richtlinie für die dynamische Skalierung erstellen) aus.
5. Wählen Sie als Richtlinientyp die Option Step Scaling und geben Sie dann einen Namen für die Richtlinie an.
6. Wählen Sie für CloudWatch Alarm Ihren Alarm aus. Wenn Sie noch keinen Alarm erstellt haben, wählen Sie Alarm erstellen und führen Sie die Schritte 4 bis 14 des vorherigen Verfahrens aus, um einen Alarm zu erstellen. CloudWatch

7. Geben Sie die Änderung der aktuellen Gruppengröße an, die diese Richtlinie vornehmen soll, wenn sie mit Take the action (Aktion ausführen) ausgeführt wird. Sie können eine bestimmte Anzahl von Instances oder einen Prozentsatz der vorhandenen Gruppengröße hinzufügen, oder die Gruppe auf eine genaue Größe festlegen.

Um beispielsweise eine Scale-Out-Richtlinie zu erstellen, die die Kapazität der Gruppe um 30 Prozent erhöht, wählen Sie `Add`, geben Sie `30` in das nächste Feld ein, und wählen Sie dann `percent of group`. Standardmäßig ist die Untergrenze dieser Schrittanpassung der Alarmschwellenwert, und die Obergrenze ist positive (+) Unendlichkeit.

8. Um einen weiteren Schritt hinzuzufügen, wählen Sie `Add step` (Schritt hinzufügen) und definieren dann den Betrag, um den skaliert werden soll, sowie die untere und obere Grenze des Schritts relativ zum Alarmschwellenwert.
9. Um eine Mindestanzahl von zu skalierenden Instances festzulegen, aktualisieren Sie das Zahlenfeld unter `Add capacity units in increments of at least` (Kapazitätseinheiten hinzufügen in Schritten von mindestens) 1 Kapazitätseinheiten.
10. (Optional) Aktualisieren Sie für Instance-Warmup den Instanz-Warmup-Wert nach Bedarf.
11. Wählen Sie `Erstellen` aus.

AWS CLI

Um eine Richtlinie zur schrittweisen Skalierung für die horizontale Skalierung (Erhöhung der Kapazität) zu erstellen, können Sie die folgenden Beispielbefehle verwenden. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

Wenn Sie die verwenden AWS CLI, erstellen Sie zunächst eine Richtlinie zur schrittweisen Skalierung, die Amazon EC2 Auto Scaling Anweisungen zur Skalierung bei steigendem Wert einer Metrik bereitstellt. Anschließend erstellen Sie den Alarm, indem Sie die zu überwachende Metrik identifizieren, den Schwellenwert für die Metrik und andere Details für die Alarme definieren und den Alarm der Skalierungsrichtlinie zuordnen.

Schritt 1: Erstellen Sie eine Richtlinie für Scale-Out

Verwenden Sie den folgenden `put-scaling-policy` Befehl, um eine schrittweise Skalierungsrichtlinie mit dem Namen `my-step-scale-out-policy` zu erstellen, deren Anpassungstyp die Kapazität der Gruppe auf der Grundlage der folgenden schrittweisen Anpassungen erhöht (unter der Annahme eines CloudWatch Alarmschwellenwerts von 60 Prozent): `PercentChangeInCapacity`

- Erhöhen Sie die Anzahl der Instances um 10 Prozent, wenn der Wert der Metrik größer oder gleich 60 Prozent, aber kleiner als 75 Prozent ist.
- Erhöhen Sie die Anzahl der Instances um 20 Prozent, wenn der Wert der Metrik größer oder gleich 75 Prozent, aber kleiner als 85 Prozent ist.
- Erhöhen Sie die Anzahl der Instances um 30 Prozent, wenn der Wert der Metrik größer oder gleich 85 Prozent ist.

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
MetricIntervalLowerBound=0.0,MetricIntervalUpperBound=15.0,ScalingAdjustment=10 \

MetricIntervalLowerBound=15.0,MetricIntervalUpperBound=25.0,ScalingAdjustment=20 \
  MetricIntervalLowerBound=25.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Richtlinie. Sie benötigen ihn, um einen CloudWatch Alarm für die Richtlinie zu erstellen.

```
{
  "PolicyARN":
  "arn:aws:autoscaling:region:123456789012:scalingPolicy:4ee9e543-86b5-4121-b53b-aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-step-scale-in-policy
}
```

Schritt 2: Erstellen Sie einen CloudWatch Alarm für den hohen Schwellenwert der Metrik

Verwenden Sie den folgenden CloudWatch [put-metric-alarm](#) Befehl, um einen Alarm zu erstellen, der die Auto Scaling Scaling-Gruppe auf der Grundlage eines durchschnittlichen CPU-Schwellenwerts von 60 Prozent für mindestens zwei aufeinanderfolgende Evaluierungsperioden von zwei Minuten vergrößert. Geben Sie zum Verwenden einer selbst erstellten Metrik den Namen der Metrik im Feld `--metric-name` und ihren Namespace im Feld `--namespace` an.

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-AddCapacity \
```

```
--metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \  
--period 120 --evaluation-periods 2 --threshold 60 \  
--comparison-operator GreaterThanOrEqualToThreshold \  
--dimensions "Name=AutoScalingGroupName,Value=my-asg" \  
--alarm-actions PolicyARN
```

Erstellen Sie eine Richtlinie zur schrittweisen Skalierung für die Skalierung

Verwenden Sie eine der folgenden Methoden, um eine schrittweise Skalierungsrichtlinie für die Skalierung für Ihre Auto Scaling-Gruppe zu erstellen:

Console

Schritt 1: Erstellen Sie einen CloudWatch Alarm für den niedrigen Schwellenwert der Metrik

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Ändern Sie, falls erforderlich, die Region. Wählen Sie auf der Navigationsleiste die Region aus, in der sich Ihre Auto-Scaling-Gruppe befindet.
3. Wählen Sie im Navigationsbereich Alarms > All alarms (Alarme > Alle Alarme) und anschließend Create alarm (Alarm erstellen) aus.
4. Wählen Sie Select metric (Metrik auswählen) aus.
5. Wählen Sie auf der Registerkarte Alle Metriken die Option EC2 und Nach Auto-Scaling-Gruppe aus und geben Sie den Namen der Auto-Scaling-Gruppe in das Suchfeld ein. Wählen Sie dann CPUUtilization und anschließend Metrik auswählen aus. Die Seite Specify metric and conditions (Metrik und Bedingungen festlegen) mit einem Diagramm und weiteren Informationen über die Metrik werden angezeigt.
6. Wählen Sie unter Period (Zeitraum) den Auswertungszeitraum für den Alarm aus, z. B. 1 Minute. Beim Auswerten des Alarms wird jeder Zeitraum in einem Datenpunkt zusammengefasst.

Note

Ein kürzerer Zeitraum erzeugt eine höhere Alarmempfindlichkeit.

7. Führen Sie unter Bedingungen die folgenden Schritte aus:
 - Wählen Sie für Threshold type (Schwellenwerttyp) die Option Static (Statisch) aus.

- Geben Sie für Whenever **CPUUtilization** is an, ob der Wert der Metrik kleiner oder kleiner als oder gleich dem Schwellenwert sein soll, ab dem der Alarm überschritten werden kann. Geben Sie dann unter than (als) den Schwellenwert ein, der den Alarm auslösen soll.

 Important

Damit ein Alarm mit einer Richtlinienskala (Metrik niedrig) verwendet werden kann, stellen Sie sicher, dass Sie nicht größer oder größer als oder gleich dem Schwellenwert wählen.

8. Führen Sie unter Zusätzliche Konfiguration die folgenden Schritte aus:

- Geben Sie unter Datenpunkte zum Alarm die Anzahl der Datenpunkte (Auswertungszeiträume) ein, während denen der Metrikwert die Schwellenbedingungen des Alarms erfüllen muss. So würde es bei zwei aufeinanderfolgenden Zeiträume von je 5 Minuten z. B. 10 Minuten dauern, den Alarmstatus auszulösen.
- Wählen Sie für Fehlende Datenbehandlung die Option Fehlende Daten als ungültig behandeln (Überschreitungsschwelle) aus. Weitere Informationen finden Sie unter [Konfiguration der Behandlung fehlender Daten durch CloudWatch Alarme](#) im CloudWatch Amazon-Benutzerhandbuch.

9. Wählen Sie Weiter aus.

Die Seite Configure actions (Konfigurieren von Aktionen) wird angezeigt.

10. Wählen Sie unter Notification (Benachrichtigung) ein Amazon-SNS-Thema aus, das benachrichtigt werden soll, wenn sich der Alarm im Zustand ALARM, OK oder INSUFFICIENT_DATA befindet.

Um zu erreichen, dass der Alarm mehrere Benachrichtigungen für den gleichen Alarmstatus oder für verschiedene Statuswerte sendet, wählen Sie Benachrichtigung hinzufügen.

Damit der Alarm keine Benachrichtigungen sendet, wählen Sie Remove (Entfernen).

11. Sie können die anderen Abschnitte der Seite Configure actions (Konfigurieren von Aktionen) leer lassen. Wenn Sie die anderen Abschnitte leer lassen, wird ein Alarm erstellt, ohne diesen einer Skalierungsrichtlinie zuzuordnen. Sie können den Alarm dann über die Amazon EC2 Auto Scaling-Konsole mit einer Skalierungsrichtlinie verknüpfen.

12. Wählen Sie Weiter aus.

13. Geben Sie einen Namen (beispielsweise `Step-Scaling-AlarmLow-RemoveCapacity`) und optional eine Beschreibung des Alarms ein. Wählen Sie anschließend Next (Weiter) aus.
14. Wählen Sie Alarm erstellen aus.

Gehen Sie wie folgt vor, um dort weiterzumachen, wo Sie nach der Erstellung Ihres CloudWatch Alarms aufgehört haben.

Schritt 2: Erstellen Sie eine Richtlinie zur schrittweisen Skalierung für die Skalierung

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Stellen Sie sicher, dass die Skalierungslimits entsprechend festgelegt sind. Wenn die gewünschte Kapazität Ihrer Gruppe beispielsweise bereits das Minimum erreicht hat, müssen Sie ein neues Minimum angeben, um skalieren zu können. Weitere Informationen finden Sie unter [Festlegen von Skalierungslimits für Ihre Auto-Scaling-Gruppe](#).
4. Wählen Sie auf der Registerkarte Automatic scaling (Automatische Skalierung) unter Dynamic scaling policies (Dynamische Skalierungsrichtlinien) die Option Create dynamic scaling policy (Richtlinie für die dynamische Skalierung erstellen) aus.
5. Wählen Sie als Richtlinientyp die Option Schrittweise Skalierung aus, und geben Sie dann einen Namen für die Richtlinie an.
6. Wählen Sie für CloudWatch Alarm Ihren Alarm aus. Wenn Sie noch keinen Alarm erstellt haben, wählen Sie Alarm erstellen und führen Sie die Schritte 4 bis 14 des vorherigen Verfahrens aus, um einen Alarm zu erstellen. CloudWatch
7. Geben Sie die Änderung der aktuellen Gruppengröße an, die diese Richtlinie vornehmen soll, wenn sie mit Take the action (Aktion ausführen) ausgeführt wird. Sie können eine bestimmte Anzahl von Instances oder einen Prozentsatz der vorhandenen Gruppengröße entfernen, oder die Gruppe auf eine genaue Größe festlegen.

Um beispielsweise eine Richtlinienskala zu erstellen, die die Kapazität der Gruppe um zwei Instanzen verringert, wählen Sie `Remove`, geben Sie 2 in das nächste Feld ein, und wählen Sie `capacity units`. Standardmäßig ist die Obergrenze dieser Schrittanpassung der Alarmschwellenwert, und die Untergrenze ist negative (-) Unendlichkeit.

- Um einen weiteren Schritt hinzuzufügen, wählen Sie Add step (Schritt hinzufügen) und definieren dann den Betrag, um den skaliert werden soll, sowie die untere und obere Grenze des Schritts relativ zum Alarmschwellenwert.
- Wählen Sie Erstellen aus.

AWS CLI

Um eine Richtlinie zur schrittweisen Skalierung für die Skalierung (Kapazität verringern) zu erstellen, können Sie die folgenden Beispielbefehle verwenden. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

Wenn Sie die verwenden AWS CLI, erstellen Sie zunächst eine Richtlinie zur schrittweisen Skalierung, die Amazon EC2 Auto Scaling Anweisungen zur Skalierung bereitstellt, wenn der Wert einer Metrik sinkt. Anschließend erstellen Sie den Alarm, indem Sie die zu überwachende Metrik identifizieren, den unteren Schwellenwert für die Metrik und andere Details für die Alarme definieren und den Alarm der Skalierungsrichtlinie zuordnen.

Schritt 1: Erstellen Sie eine Richtlinie für die Skalierung

Verwenden Sie den folgenden [put-scaling-policy](#) Befehl, um eine schrittweise Skalierungsrichtlinie mit dem Namen zu erstellen `my-step-scale-in-policy`, deren Anpassungstyp die Kapazität der Gruppe um 2 Instanzen verringert, wenn der zugehörige CloudWatch Alarm den unteren Schwellenwert der Metrik überschreitet. `ChangeInCapacity`

```
aws autoscaling put-scaling-policy \  
  --auto-scaling-group-name my-asg \  
  --policy-name my-step-scale-in-policy \  
  --policy-type StepScaling \  
  --adjustment-type ChangeInCapacity \  
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Richtlinie. Sie benötigen ihn, um den CloudWatch Alarm für die Richtlinie zu erstellen.

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-step-scale-out-policy  
}
```

Schritt 2: Erstellen Sie einen CloudWatch Alarm für den unteren Schwellenwert der Metrik

Verwenden Sie den folgenden CloudWatch [put-metric-alarm](#) Befehl, um einen Alarm zu erstellen, der die Größe der Auto Scaling Scaling-Gruppe auf der Grundlage eines durchschnittlichen CPU-Schwellenwerts von 40 Prozent für mindestens zwei aufeinanderfolgende Evaluierungsperioden von zwei Minuten verringert. Geben Sie zum Verwenden einer selbst erstellten Metrik den Namen der Metrik im Feld `--metric-name` und ihren Namespace im Feld `--namespace` an.

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmLow-RemoveCapacity \  
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \  
  --period 120 --evaluation-periods 2 --threshold 40 \  
  --comparison-operator LessThanOrEqualToThreshold \  
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \  
  --alarm-actions PolicyARN
```

Einfache Skalierungsrichtlinien

Die folgenden Beispiele zeigen, wie Sie CLI-Befehle verwenden können, um einfache Skalierungsrichtlinien zu erstellen. Sie bleiben in diesem Dokument als Referenz für alle Kunden, die sie verwenden möchten, enthalten. Wir empfehlen jedoch, stattdessen Target-Tracking- oder Step-Scaling-Richtlinien zu verwenden.

Ähnlich wie bei Richtlinien zur schrittweisen Skalierung müssen Sie bei einfachen Skalierungsrichtlinien CloudWatch Alarme für Ihre Skalierungsrichtlinien erstellen. In den Richtlinien, die Sie erstellen, müssen Sie auch definieren, ob und wie viele Instanzen hinzugefügt oder entfernt werden sollen, oder die Gruppe auf eine exakte Größe festlegen.

Einer der Hauptunterschiede zwischen Step Scaling-Richtlinien und einfachen Skalierungsrichtlinien sind die schrittweisen Anpassungen, die Sie mit Step Scaling-Richtlinien erhalten. Bei der schrittweisen Skalierung können Sie auf der Grundlage der von Ihnen angegebenen schrittweisen Anpassungen größere oder kleinere Änderungen an der Gruppengröße vornehmen.

Eine einfache Skalierungsrichtlinie muss außerdem warten, bis eine laufende Skalierungsaktivität oder ein Ersatz für eine Integritätsprüfung abgeschlossen ist und eine [Abklingzeit abgelaufen](#) ist, bevor sie auf weitere Alarme reagiert. Im Gegensatz dazu reagiert die Richtlinie bei der schrittweisen Skalierung weiterhin auf zusätzliche Alarme, selbst wenn eine Skalierungsaktivität oder ein Ersatz für einen Gesundheitscheck im Gange ist. Das bedeutet, dass Amazon EC2 Auto Scaling alle Alarmverletzungen bewertet, sobald es die Alarmmeldungen empfängt. Aus diesem Grund empfehlen

wir, stattdessen Richtlinien zur schrittweisen Skalierung zu verwenden, auch wenn Sie nur eine einzige Skalierungsanpassung haben.

Amazon EC2 Auto Scaling hat ursprünglich nur einfache Skalierungsrichtlinien unterstützt. Wenn Sie Ihre Skalierungsrichtlinie vor der Einführung von Richtlinien zur Zielverfolgung und schrittweisen Skalierung erstellt haben, wird Ihre Richtlinie als einfache Skalierungsrichtlinie behandelt.

Erstellen Sie eine einfache Skalierungsrichtlinie für Scale-Out

Verwenden Sie den folgenden [put-scaling-policy](#) Befehl, um eine einfache Skalierungsrichtlinie mit dem Namen `my-simple-scale-out-policy`, mit einem Anpassungstyp zu erstellen `PercentChangeInCapacity`, der die Kapazität der Gruppe um 30 Prozent erhöht, wenn der zugehörige CloudWatch Alarm den oberen Schwellenwert der Metrik überschreitet.

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \  
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \  
  --adjustment-type PercentChangeInCapacity
```

Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Richtlinie. Sie benötigen ihn, um den CloudWatch Alarm für die Richtlinie zu erstellen.

Erstellen Sie eine einfache Skalierungsrichtlinie für die Skalierung

Verwenden Sie den folgenden [put-scaling-policy](#) Befehl, um eine einfache Skalierungsrichtlinie mit dem Namen `my-simple-scale-in-policy`, mit einem Anpassungstyp zu erstellen `ChangeInCapacity`, der die Kapazität der Gruppe um eine Instanz verringert, wenn der zugehörige CloudWatch Alarm den unteren Schwellenwert der Metrik überschreitet.

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \  
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \  
  --adjustment-type ChangeInCapacity --cooldown 180
```

Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Richtlinie. Sie benötigen ihn, um den CloudWatch Alarm für die Richtlinie zu erstellen.

Skalierungsruhephasen für Amazon EC2 Auto Scaling

Important

Als bewährte Methode empfehlen wir Ihnen, keine einfachen Skalierungsrichtlinien und Skalierungs-Cooldowns zu verwenden. Eine Skalierungsrichtlinie für die Zielverfolgung oder

eine Richtlinie für die schrittweise Skalierung ist besser für die Skalierung der Leistung. Für eine Skalierungsrichtlinie, welche die Größe Ihrer Auto-Scaling-Gruppe proportional verändert, wenn der Wert der Skalierungsmetrik ab- oder zunimmt, sollte eine [Zielverfolgung](#) statt einer einfachen oder Schrittskalierung verwendet werden.

Wenn Sie einfache Skalierungsrichtlinien für Ihre Auto-Scaling-Gruppe erstellen, empfehlen wir, dass Sie gleichzeitig die Skalierungs-Ruhephase konfigurieren.

Nachdem Ihre Auto-Scaling-Gruppe Instances gestartet oder beendet hat, wartet sie auf das Ende der Ruhephase, bevor weitere Skalierungsaktivitäten gestartet werden können, die durch einfache Skalierungsrichtlinien initiiert werden. Der Zweck der Ruhephase besteht darin, dass sich Ihre Auto-Scaling-Gruppe stabilisiert und verhindert, dass weitere Instances gestartet oder beendet werden, bevor die Auswirkungen der vorherigen Skalierungsaktivität sichtbar sind.

Angenommen, eine einfache Skalierungsrichtlinie für die CPU-Auslastung empfiehlt, zwei Instances zu starten. Amazon EC2 Auto Scaling startet zwei Instances und unterbricht dann die Skalierungen bis zum Ende der Ruhephase. Nach Ende der Ruhephase können alle Skalierungen, die durch einfache Skalierungsrichtlinien ausgelöst werden, fortgesetzt werden. Wenn die CPU-Auslastung erneut die Alarmhöhe verletzt, skaliert die Auto-Scaling-Gruppe erneut, und die Ruhephase tritt erneut in Kraft. Wenn jedoch zwei Instances ausgereicht haben, um den Metrikwert zu verringern, bleibt die aktuelle Größe der Gruppe erhalten.

Inhalt

- [Überlegungen](#)
- [Lebenszyklus-Hooks können zusätzliche Verzögerungen verursachen](#)
- [Ändern der standardmäßigen Ruhephase](#)
- [Festlegen einer Ruhephase für bestimmte einfache Skalierungsrichtlinien](#)

Überlegungen

Die folgenden Überlegungen gelten bei der Arbeit mit einfachen Skalierungsrichtlinien und Skalierung von Cooldowns:

- Richtlinien zur Zielverfolgung und zur Schrittskalierung können sofort eine Aufskalierungs-Aktivität initiieren, ohne auf das Ende der Ruhephase zu warten. Stattdessen haben die einzelnen Instances immer dann, wenn Ihre Auto Scaling Scaling-Gruppe Instances startet, eine Aufwärmphase.

Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

- Während einer Scale-Out-Aktivität werden alle Scale-In-Aktivitäten zur Zielverfolgung und schrittweisen Skalierung blockiert, bis die Instances das Warmlaufen abgeschlossen haben. Scale-in-Aktivitäten können sich auch verzögern, wenn sich eine Auto Scaling Scaling-Gruppe in einer Abklingzeit befindet.
- Wenn eine geplante Aktion während einer Ruhephase zum geplanten Zeitpunkt beginnt, kann sie umgehend eine Skalierung auslösen, ohne das Ende der Ruhephase abzuwarten.
- Wenn eine Instance fehlerhaft wird, wartet Amazon EC2 Auto Scaling nicht auf das Ende der Ruhephase, bevor die Instance ersetzt wird.
- Wenn mehrere Instances launchen oder beenden, beginnt die Ruhephase (sei es die standardmäßige oder die Skalierungsrichtlinienspezifische Ruhephase) nach erfolgreichem Start oder erfolgter Beendigung der letzten Instance.
- Wenn Sie Ihre Auto-Scaling-Gruppe manuell skalieren, wird standardmäßig nicht auf das Ende eines Cooldown-Vorgangs gewartet. Sie können dieses Verhalten jedoch außer Kraft setzen und die standardmäßige Abklingzeit beibehalten, wenn Sie das AWS CLI oder ein SDK für die manuelle Skalierung verwenden.
- Standardmäßig wartet Elastic Load Balancing 300 Sekunden, um den Abmeldeprozess (Connection Draining) abzuschließen. Wenn sich die Gruppe hinter einer Elastic Load Balancing-Load-Balancer befindet, wartet sie, bis sich die abschließenden Instances abmelden, bevor die Ruhephase beginnt.

Lebenszyklus-Hooks können zusätzliche Verzögerungen verursachen

Wenn ein [Lebenszyklus-Hook](#) aufgerufen wird, beginnt die Ruhephase, nachdem Sie die Lebenszyklus-Aktion abgeschlossen haben oder nach Ende des Timeout-Zeitraums. Angenommen, eine Auto-Scaling-Gruppe besitzt einen Lebenszyklus-Hook für den Instance-Start. Steigt die Auslastung der Anwendung, startet die Gruppe eine Instance, um die Kapazität zu erhöhen. Da es einen Lebenszyklus-Hook gibt, wird die Instance in einen Wartezustand versetzt und Skalierungen aufgrund einfacher Skalierungsrichtlinien werden angehalten. Die Ruhephase beginnt, wenn die Instance in den Status `InService` versetzt wird. Nach Ablauf der Ruhephase werden einfache Skalierungsrichtlinien wieder aufgenommen.

Wenn Elastic Load Balancing für die Skalierung aktiviert ist, beginnt die Abklingzeit, wenn die für die Kündigung ausgewählte Instance mit dem Verbindungsabbau beginnt (Abmeldeverzögerung). Die

Abklingzeit wartet nicht darauf, dass der Verbindungsabbau abgeschlossen ist oder der Lifecycle-Hook seine Aktion abgeschlossen hat. Das bedeutet, dass alle Skalierungsaktivitäten, die auf einfache Skalierungsrichtlinien zurückzuführen sind, wieder aufgenommen werden können, sobald sich das Ergebnis der Skalierung in der Kapazität der Gruppe niederschlägt. Andernfalls erhöht das Warten auf den Abschluss aller drei Aktivitäten (Connection-Draining, Lebenszyklus-Hook und Ruhephase) die Zeit, welche die Auto-Scaling-Gruppe benötigt, um die Skalierung zu unterbrechen.

Ändern der standardmäßigen Ruhephase

Sie können den standardmäßige Cooldown-Vorgang nicht festlegen, wenn Sie zunächst eine Auto-Scaling-Gruppe in der Scaling-Konsole von Amazon EC2 Auto Scaling erstellen. Standardmäßig ist diese Ruhephase auf 300 Sekunden (5 Minuten) festgelegt. Bei Bedarf können Sie dies aktualisieren, nachdem die Gruppe erstellt wurde.

So ändern Sie die standardmäßige Ruhephase (Konsole)

Nach dem Erstellen der Auto-Scaling-Gruppe wählen Sie auf der Registerkarte Details, *Advances configurations* (Erweiterte Konfigurationen) und dann *Edit* (Bearbeiten) aus. Für *Default cooldown* (Standardmäßige Ruhephase/Cooldown-Vorgang) wählen Sie die gewünschte Zeit auf der Grundlage der Startzeit für Ihre Instance oder anderer Anwendungsanforderungen aus.

Ändern der standardmäßigen Ruhephase (AWS CLI)

Verwenden Sie die folgenden Befehle, um die standardmäßige Ruhephase für neue oder vorhandene Auto-Scaling-Gruppen zu ändern. Wenn die standardmäßige Ruhephase nicht definiert ist, wird der Standardwert von 300 Sekunden verwendet.

- [create-auto-scaling-group](#)
- [update-auto-scaling-group](#)

Verwenden Sie den [describe-auto-scaling-groups](#) Befehl, um den Wert der Standard-Abklingzeit zu bestätigen.

Festlegen einer Ruhephase für bestimmte einfache Skalierungsrichtlinien

Standardmäßig verwenden alle einfachen Skalierungsrichtlinien die für die Auto-Scaling-Gruppe definierte Standard-Ruhephase. Wenn Sie eine Ruhephase für bestimmte einfache Skalierungsrichtlinien angeben möchten, verwenden Sie den optionalen Parameter für die

Ruhephase beim Erstellen oder Aktualisieren der Richtlinie. Wenn für eine Richtlinie eine Ruhephase angegeben wird, überschreibt sie die standardmäßige Ruhephase.

Eine allgemeine Verwendung für eine skalierungsrichtlinienspezifische Abklingzeit ist die Verwendung einer Richtlinienskala. Da diese Richtlinie Instances beendet, benötigt Amazon EC2 Auto Scaling weniger Zeit, um zu ermitteln, ob weitere Instances zu beenden sind. Die Beendigung von Instances sollte viel schneller als der Start von Instances erfolgen. Die standardmäßige Ruhephase von 300 Sekunden ist daher zu lang. In diesem Fall kann eine skalierbare, richtlinienspezifische Abklingzeit mit einem niedrigeren Wert für Ihre politische Skalierung Ihnen helfen, die Kosten zu senken, da die Gruppe schneller skalieren kann.

Um einfache Skalierungsrichtlinien in der Konsole zu erstellen oder zu aktualisieren, wählen Sie die Auto Scaling (Automatische Skalierung) nachdem Sie die Gruppe erstellt haben. Verwenden Sie den Befehl, um einfache Skalierungsrichtlinien mithilfe von zu erstellen oder zu aktualisieren. [AWS CLI `put-scaling-policy`](#) Weitere Informationen finden Sie unter [Schrittweise und einfache Skalierungsrichtlinien](#).

Skalierungsrichtlinie basierend auf Amazon SQS

Important

Die folgenden Informationen und Schritte zeigen Ihnen, wie Sie den Amazon SQS-Warteschlangen-Backlog pro Instance anhand des `ApproximateNumberOfMessages` Queue-Attributs berechnen, bevor Sie ihn als benutzerdefinierte Metrik für veröffentlichen. CloudWatch Sie können jetzt jedoch die Kosten und den Aufwand für die Veröffentlichung Ihrer eigenen Metrik sparen, indem Sie metrische Berechnungen verwenden. Weitere Informationen finden Sie unter [Erstellen Sie mithilfe metrischer Mathematik eine Skalierungsrichtlinie für die Zielverfolgung](#).

Sie können Ihre Auto Scaling Scaling-Gruppe als Reaktion auf Änderungen der Systemlast in einer Amazon Simple Queue Service (Amazon SQS) -Warteschlange skalieren. Weitere Informationen zur Verwendung von Amazon SQS finden Sie im [Amazon Simple Queue Service-Entwicklerhandbuch](#).

Es gibt einige Szenarien, in denen Sie als Reaktion auf Aktivitäten in einer Amazon SQS-Warteschlange eine Skalierung erwägen könnten. Angenommen, Sie haben eine Webanwendung, mit der Benutzer Bilder hochladen und online verwenden können. In diesem Szenario erfordert jedes Image eine Größenanpassung und Codierung, bevor es veröffentlicht werden kann. Die App wird auf EC2-Instances einer Auto-Scaling-Gruppe ausgeführt und ist für die Verarbeitung

Ihrer typischen Upload-Raten konfiguriert. Fehlerhafte Instances werden beendet und ersetzt, um stets dieselbe Anzahl an Instances zu nutzen. Die App platziert die Raw-Bitmap-Daten der Bilder in eine SQS-Warteschlange für die Verarbeitung. Sie verarbeitet die Bilder und veröffentlicht die verarbeiteten Bilder, wo sie von Benutzern angezeigt werden können. Die Architektur für dieses Szenario funktioniert gut, wenn die Anzahl der Image-Uploads nicht im Laufe der Zeit variiert. Wenn sich die Anzahl der Uploads jedoch mit der Zeit ändert, könnten Sie eine dynamische Skalierung in Betracht ziehen, um die Kapazität Ihrer Auto-Scaling-Gruppe zu skalieren.

Inhalt

- [Zielverfolgung mit der richtigen Metrik verwenden](#)
- [Einschränkungen](#)
- [Konfigurieren der Skalierung basierend auf Amazon SQS](#)
- [Amazon SQS und Instance-Skalierungsschutz](#)

Zielverfolgung mit der richtigen Metrik verwenden

Wenn Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung verwenden, die auf einer benutzerdefinierten Amazon SQS-Warteschlangenmetrik basiert, kann die dynamische Skalierung eine effektivere Anpassung an die Bedarfskurve Ihrer Anwendung vornehmen. Weitere Informationen zum Auswählen von Metriken für die Zielverfolgung finden Sie unter [Auswahl von Metriken](#).

Das Problem bei der Verwendung einer CloudWatch Amazon SQS-Metrik wie `ApproximateNumberOfMessagesVisible` für die Zielverfolgung besteht darin, dass sich die Anzahl der Nachrichten in der Warteschlange möglicherweise nicht proportional zur Größe der Auto Scaling Scaling-Gruppe ändert, die Nachrichten aus der Warteschlange verarbeitet. Das liegt daran, dass die Anzahl der Nachrichten in der SQS-Warteschlange nicht alleine die Anzahl der erforderlichen Instances definiert. Tatsächlich kann die Anzahl der Instances in Ihrer Auto-Scaling-Gruppe von mehreren Faktoren abhängen, einschließlich der für die Verarbeitung einer Nachricht benötigten Zeit und der akzeptablen zeitlichen Latenz (Warteschlangenverzögerung).

Die Lösung besteht darin, eine Rückstand pro Instance-Metrik zu verwenden, deren Zielwert der zulässige Rückstand pro Instance ist, der aufrechterhalten werden soll. Sie können diese Zahlen wie folgt berechnen:

- Rückstand pro Instance: Um den Rückstand pro Instance zu berechnen, verwenden Sie zunächst das Warteschlangenattribut `ApproximateNumberOfMessages`, um die Länge der SQS-Warteschlange (Anzahl der Nachrichten, die zum Abrufen aus der Warteschlange verfügbar

sind) zu bestimmen. Dividieren Sie diese Zahl durch die laufende Kapazität der Flotte, welche bei einer Auto-Scaling-Gruppe die Anzahl der Instances mit dem Status `InService` ist, um so den Rückstand pro Instance zu erhalten.

- Zulässiger Rückstand pro Instance: Um Ihren Zielwert zu berechnen, bestimmen Sie zunächst, welche zeitliche Latenz Ihre Anwendung akzeptieren kann. Teilen Sie diesen akzeptablen Latenzwert dann durch die durchschnittliche Zeit, die eine EC2-Instance für die Verarbeitung einer Nachricht benötigt.

Ein Beispiel: Angenommen, Sie verfügen über eine Auto-Scaling-Gruppe mit 10 Instances und die Anzahl sichtbarer Nachrichten in der Warteschlange (`ApproximateNumberOfMessages`) ist 1 500. Wenn die durchschnittliche Verarbeitungszeit pro Nachricht 0,1 Sekunden beträgt und die längste akzeptable Latenz 10 Sekunden ist, dann ist der zulässige Rückstand pro Instance $10/0,1$, was 100 Nachrichten entspricht. Dies bedeutet, dass 100 der Zielwert für Ihre Ziel-Nachverfolgungsrichtlinie ist. Wenn der Rückstand pro Instance den Zielwert erreicht, wird ein Aufskalierungsereignis ausgelöst. Da der Rückstand pro Instance bereits bei 150 Nachrichten liegt ($1\ 500$ Nachrichten / 10 Instances), wird Ihre Gruppe um fünf Instances aufskaliert, um die Proportion zum Zielwert beizubehalten.

Die folgenden Verfahren veranschaulichen, wie Sie die benutzerdefinierte Metrik veröffentlichen und die Skalierungsrichtlinie für die Ziel-Nachverfolgung erstellen, die Ihre Auto-Scaling-Gruppe zum Skalieren basierend auf diesen Berechnungen konfiguriert.

Important

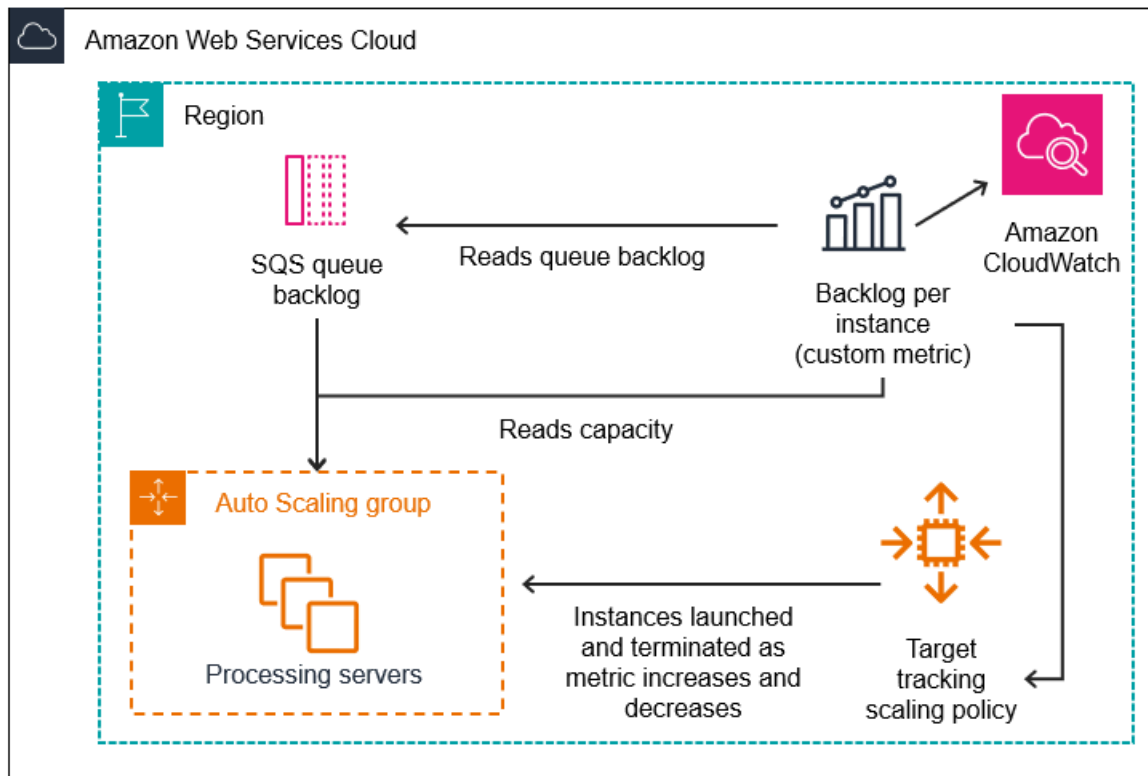
Denken Sie daran, zur Kostensenkung stattdessen metrische Berechnungen zu verwenden. Weitere Informationen finden Sie unter [Erstellen Sie mithilfe metrischer Mathematik eine Skalierungsrichtlinie für die Zielverfolgung](#).

Es gibt drei Hauptkomponenten dieser Konfiguration:

- Eine Auto-Scaling-Gruppe zur Verwaltung von EC2-Instances für die Verarbeitung von Nachrichten aus einer SQS-Warteschlange.
- Eine benutzerdefinierte Metrik zum Senden an Amazon CloudWatch, die die Anzahl der Nachrichten in der Warteschlange pro EC2-Instance in der Auto Scaling Scaling-Gruppe misst.

- Eine Zielverfolgungsrichtlinie, die Ihre Auto Scaling Scaling-Gruppe so konfiguriert, dass sie auf der Grundlage der benutzerdefinierten Metrik und eines festgelegten Zielwerts skaliert. CloudWatch Alarme rufen die Skalierungsrichtlinie auf.

Das folgende Diagramm illustriert die Architektur dieser Konfiguration.



Einschränkungen

Sie müssen das AWS CLI oder ein SDK verwenden, um Ihre benutzerdefinierte Metrik zu CloudWatch veröffentlichen. Anschließend können Sie Ihre Metrik mit dem überwachen AWS-Managementkonsole.

In den folgenden Abschnitten verwenden Sie die AWS CLI für die Aufgaben, die Sie ausführen müssen. Um beispielsweise metrische Daten abzurufen, die die aktuelle Nutzung der Warteschlange widerspiegeln, verwenden Sie den [get-queue-attributes](#) SQS-Befehl.

Konfigurieren der Skalierung basierend auf Amazon SQS

Das folgende Verfahren beschreibt, wie die automatische Skalierung auf Basis von Amazon SQS konfiguriert wird. Sie erfahren, wie Sie eine CloudWatch benutzerdefinierte Metrik erstellen, wie Sie

mithilfe von eine Richtlinie zur Zielverfolgung einrichten und wie Sie Ihre Konfiguration testen. AWS CLI

Bevor Sie beginnen, stellen Sie sicher, dass Sie das AWS CLI [installiert](#) und [konfiguriert](#) haben. Außerdem müssen Sie über eine Amazon SQS SQS-Warteschlange verfügen, die Sie verwenden können. Bei den folgenden Aufgaben wird davon ausgegangen, dass Sie bereits über eine Warteschlange (Standard oder FIFO), eine Auto Scaling Scaling-Gruppe und EC2-Instances verfügen, auf denen die Anwendung ausgeführt wird, die die Warteschlange verwendet.

Weitere Informationen zu Amazon SQS, finden Sie unter [Entwicklerhandbuch für Amazon Simple Queue Service](#).

Aufgaben

- [Schritt 1: Erstellen Sie eine benutzerdefinierte Metrik CloudWatch](#)
- [Schritt 2: Erstellen einer Skalierungsrichtlinie für die Ziel-Nachverfolgung](#)
- [Schritt 3: Testen Ihrer Skalierungsrichtlinie](#)

Schritt 1: Erstellen Sie eine benutzerdefinierte Metrik CloudWatch

Eine benutzerdefinierte Metrik wird mit einem Metriknamen und Namespace Ihrer Wahl definiert. Namespaces für benutzerdefinierte Metriken können nicht mit `AWS/` beginnen. Weitere Informationen zum Veröffentlichen von benutzerdefinierten Metriken finden Sie unter dem Thema [Veröffentlichen von benutzerdefinierten Kennzahlen](#) im CloudWatch Amazon-Benutzerhandbuch.

Gehen Sie wie folgt vor, um die benutzerdefinierte Metrik zu erstellen, indem Sie zunächst die Informationen aus Ihrem AWS Konto lesen. Berechnen Sie dann den Rückstand pro Instance-Metrik, wie in einem früheren Abschnitt empfohlen. Veröffentlichen Sie diese Zahl abschließend mit CloudWatch einer Genauigkeit von 1 Minute. Wenn möglich, empfehlen wir dringend, anhand von Metriken mit einer Granularität von 1 Minute zu skalieren, um eine schnellere Reaktion auf Änderungen der Systemauslastung zu gewährleisten.

Um eine CloudWatch benutzerdefinierte Metrik zu erstellen ()AWS CLI

1. Fordern Sie mit dem SQS-Befehl [get-queue-attributes](#) die Anzahl der in der Warteschlange wartenden Nachrichten an (`ApproximateNumberOfMessages`):

```
aws sqs get-queue-attributes --queue-url https://  
sqs.region.amazonaws.com/123456789/MyQueue \
```

```
--attribute-names ApproximateNumberOfMessages
```

2. Fordern Sie mit dem Befehl [describe-auto-scaling-groups](#) die laufende Kapazität der Gruppe an, wobei es sich um die Anzahl von Instances mit dem Lebenszyklusstatus `InService` handelt. Dieser Befehl gibt die Instances einer Auto-Scaling-Gruppe zusammen mit ihren Lebenszyklusstatus zurück.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

3. Berechnen Sie den Rückstand pro Instance, indem Sie die ungefähre Anzahl der Nachrichten, die für den Abruf aus der Warteschlange verfügbar sind, durch die laufende Kapazität der Gruppe dividieren.
4. Erstellen Sie ein Skript, das jede Minute ausgeführt wird, um den Backlog-Wert pro Instanz abzurufen und ihn in einer CloudWatch benutzerdefinierten Metrik zu veröffentlichen. Beim Veröffentlichen einer benutzerdefinierten Metrik geben Sie den Namen, den Namespace, die Einheit, den Wert und null oder mehr Dimensionen für die Metrik an. Dimensionen bestehen aus einem Dimensionsnamen und einem Dimensionswert.

Um Ihre benutzerdefinierte Metrik zu veröffentlichen, ersetzen Sie Platzhalterwerte in *italics* durch Ihren bevorzugten Metriknamen, den Wert der Metrik, einen Namespace (sofern er nicht mit "AWS," beginnt) und Dimensionen (optional) und führen Sie dann den folgenden Befehl aus.

[put-metric-data](#)

```
aws cloudwatch put-metric-data --metric-name MyBacklogPerInstance --  
namespace MyNamespace \  
  --unit None --value 20 --  
dimensions MyOptionalMetricDimensionName=MyOptionalMetricDimensionValue
```

Nachdem Ihre Anwendung die gewünschte Metrik ausgegeben hat, werden die Daten an CloudWatch gesendet. Die Metrik ist in der CloudWatch Konsole sichtbar. Sie können darauf zugreifen, indem Sie sich bei der anmelden AWS-Managementkonsole und zu der CloudWatch Seite navigieren. Anschließend können Sie die Metrik anzeigen, indem Sie zur Seite der Metriken navigieren oder im Suchfeld danach suchen. Informationen zum Anzeigen von Metriken finden Sie unter [Verfügbare Metriken anzeigen](#) im CloudWatch Amazon-Benutzerhandbuch.

Schritt 2: Erstellen einer Skalierungsrichtlinie für die Ziel-Nachverfolgung

Die von Ihnen erstellte Metrik kann jetzt zu einer Skalierungsrichtlinie für die Zielnachverfolgung hinzugefügt werden.

So erstellen Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung (AWS CLI)

1. Verwenden Sie den folgenden `cat`-Befehl, um einen Zielwert für Ihre Skalierungsrichtlinie und eine benutzerdefinierte Metrikspezifikation in einer JSON-Datei namens `config.json` in Ihrem Stammverzeichnis zu speichern. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen. Berechnen Sie für den `TargetValue` die Metrik für den akzeptablen Rückstand pro Instance und geben Sie sie hier ein. Basieren Sie die Berechnung dieser Zahl auf einem normalen Latenzwert und teilen Sie ihn durch die durchschnittliche Zeit, die für die Verarbeitung einer Nachricht benötigt wird, wie in einem vorherigen Abschnitt beschrieben.

Wenn Sie keine Dimensionen für die Metrik angegeben haben, die Sie in Schritt 1 erstellt haben, nehmen Sie keine Dimensionen in die benutzerdefinierte Metrikspezifikation auf.

```
$ cat ~/config.json
{
  "TargetValue":100,
  "CustomizedMetricSpecification":{
    "MetricName":"MyBacklogPerInstance",
    "Namespace":"MyNamespace",
    "Dimensions":[
      {
        "Name":"MyOptionalMetricDimensionName",
        "Value":"MyOptionalMetricDimensionValue"
      }
    ],
    "Statistic":"Average",
    "Unit":"None"
  }
}
```

2. Verwenden Sie den [put-scaling-policy](#)-Befehl zusammen mit der Datei `config.json`, die Sie im vorherigen Schritt erstellt haben, um Ihre Skalierungsrichtlinie zu erstellen:

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
```

```
--target-tracking-configuration file://~/config.json
```

Dabei werden zwei Alarme erstellt: einer für die Aufwärtsskalierung und einer für die Abwärtsskalierung. Es gibt auch den Amazon-Ressourcennamen (ARN) der Richtlinie zurück CloudWatch, mit der die CloudWatch Skalierung aufgerufen wird, wenn der metrische Schwellenwert überschritten wird.

Schritt 3: Testen Ihrer Skalierungsrichtlinie

Wenn die Einrichtung abgeschlossen ist, überprüfen Sie, ob Ihre Skalierungsrichtlinie funktioniert. Sie testen sie, indem Sie die Anzahl der Nachrichten in der SQS-Warteschlange vergrößern und dann prüfen, ob die Auto-Scaling-Gruppe eine weitere EC2-Instance gestartet hat. Gleichermaßen testen Sie sie, indem Sie die Anzahl der Nachrichten in der SQS-Warteschlange verkleinern und dann prüfen, ob die Auto-Scaling-Gruppe eine EC2-Instance beendet hat.

So testen Sie die Funktion für die horizontale Skalierung nach oben

1. Folgen Sie den Schritten unter [Erstellen einer Amazon SQS SQS-Standardwarteschlange und Senden einer Nachricht](#) oder [Erstellen einer Amazon SQS SQS-FIFO-Warteschlange und Senden einer Nachricht](#), um Nachrichten zu Ihrer Warteschlange hinzuzufügen. Stellen Sie sicher, dass Sie die Anzahl der Nachrichten in der Warteschlange so erhöht haben, dass die Metrik für den Rückstand pro Instance den Zielwert überschreitet.

Es kann einige Minuten dauern, bis Ihre Änderungen den Alarm auslösen.

2. Verwenden Sie den Befehl [describe-auto-scaling-groups](#), um zu prüfen, ob die Gruppe eine Instance gestartet hat:

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Um die Funktion der Waage zu testen

1. Folgen Sie den Schritten unter [Nachricht empfangen und löschen \(Konsole\)](#), um Nachrichten aus der Warteschlange zu löschen. Stellen Sie sicher, dass Sie die Anzahl der Nachrichten in der Warteschlange so verringert haben, dass die Metrik für den Rückstand pro Instance den Zielwert unterschreitet.

Es kann einige Minuten dauern, bis Ihre Änderungen den Alarm auslösen.

2. Verwenden Sie den [describe-auto-scaling-groups](#)-Befehl, um zu prüfen, ob die Gruppe eine Instance beendet hat:

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Amazon SQS und Instance-Skalierungsschutz

Sofern beim Beenden einer Instance noch nicht verarbeitete Nachrichten vorhanden sind, werden diese an die SQS-Warteschlange zurückgegeben und können von einer anderen ausgeführten Instance verarbeitet werden. Für Anwendungen, in denen Aufgaben mit langer Ausführung ausgeführt werden, können Sie optional den Instance-Scale-in-Schutz verwenden, um die Kontrolle darüber zu haben, welche Warteschlangen-Worker beendet werden, wenn Ihre Auto-Scaling-Gruppe skaliert wird.

Der folgende Pseudocode zeigt eine Möglichkeit zum Schutz langjähriger, warteschlangengesteuerter Worker-Prozesse vor Scale-In-Beendigung.

```
while (true)
{
    SetInstanceProtection(False);
    Work = GetNextWorkUnit();
    SetInstanceProtection(True);
    ProcessWorkUnit(Work);
    SetInstanceProtection(False);
}
```

Weitere Informationen finden Sie unter [Gestalten Sie Ihre Anwendungen so, dass sie die Instance-Kündigung ordnungsgemäß handhaben](#).

Eine Skalierung für eine Auto-Scaling-Gruppe überprüfen

Im Abschnitt Amazon EC2 Auto Scaling der Amazon-EC2-Konsole können Sie über Activity history (Aktivitätsverlauf) für eine Auto-Scaling-Gruppe den aktuellen Status einer laufenden Skalierungsaktivität einsehen. Wenn die Skalierungsaktivität abgeschlossen ist, können Sie sehen, ob sie erfolgreich war oder nicht. Dies ist besonders nützlich, wenn Sie Auto-Scaling-Gruppen erstellen oder Skalierungsbedingungen zu bestehenden Gruppen hinzufügen.

Wenn Sie eine Zielverfolgungs-, Stufen- oder einfache Skalierungsrichtlinie zu Ihrer Auto-Scaling-Gruppe hinzufügen, beginnt Amazon EC2 Auto Scaling sofort mit der Auswertung der Richtlinie

anhand der Metrik. Der Metrik-Alarm geht in den ALARM-Zustand, wenn die Metrik für eine bestimmte Anzahl von Auswertungszeiträumen den Schwellenwert überschreitet. Das bedeutet, dass eine Skalierungsrichtlinie kurz nach ihrer Erstellung zu einer Skalierungsaktivität führen kann. Nachdem Amazon EC2 Auto Scaling die gewünschte Kapazität als Reaktion auf eine Skalierungsrichtlinie angepasst hat, können Sie die Skalierungsaktivität in Ihrem Konto überprüfen. Wenn Sie eine E-Mail-Benachrichtigung von Amazon EC2 Auto Scaling erhalten möchten, die Sie über eine Skalierungsaktivität informiert, folgen Sie den Anweisungen in [Amazon SNS SNS-Benachrichtigungsoptionen für Amazon EC2 Auto Scaling](#).

Tip

Im folgenden Verfahren sehen Sie sich die Abschnitte Activity history (Verlauf der Aktivität) und Instances (Instances) für die Auto-Scaling-Gruppe an. In beiden sollten die benannten Spalten bereits angezeigt werden. Um ausgeblendete Spalten anzuzeigen oder die Anzahl der angezeigten Zeilen zu ändern, wählen Sie das Zahnradsymbol in der oberen rechten Ecke jedes Abschnitts, um die Einstellungen zu öffnen, die Einstellungen nach Bedarf zu aktualisieren und Confirm (Bestätigen) auszuwählen.

So zeigen Sie die Skalierungsaktivitäten für eine Auto-Scaling-Gruppe an (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die Region aus, in der sich Ihre Auto-Scaling-Gruppe befindet.
3. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Auf der Registerkarte Activity (Aktivität) unter Activity history (Aktivitätsverlauf) zeigt die Spalte Status an, ob Ihre Auto-Scaling-Gruppe-Instances erfolgreich gestartet oder beendet hat oder ob die Skalierungsaktivität noch im Gange ist.
5. Wenn Sie viele Skalierungen haben, können Sie auf das Symbol > am oberen Rand des Aktivitätsverlaufs klicken, um die nächste Seite der Skalierungen anzuzeigen.
6. Auf der Registerkarte Instance management (Instance-Verwaltung) wird unter Instances in der Spalte Lifecycle (Lebenszyklus) der Zustand Ihrer Instances angezeigt. Nach dem Start der Instance und dem Ende der Lebenszyklus-Hooks ändert sich ihr Lebenszyklusstatus

in InService. In der Spalte Health status (Zustandsstatus) wird Ihnen das Ergebnis der Zustandsprüfung Ihrer EC2-Instance angezeigt.

So zeigen Sie die Skalierungsaktivitäten für eine Auto-Scaling-Gruppe an (AWS CLI)

Verwenden Sie den folgenden [describe-scaling-activities](#)-Befehl.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

Es folgt eine Beispielausgabe.

Skalierungsaktivitäten werden nach Startzeit sortiert. Die noch laufenden Aktivitäten werden zuerst beschrieben.

```
{
  "Activities": [
    {
      "ActivityId": "5e3a1f47-2309-415c-bfd8-35aa06300799",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-06c4794c2499af1df",
      "Cause": "At 2020-02-11T18:34:10Z a monitor alarm TargetTracking-my-asg-AlarmLow-b9376cab-18a7-4385-920c-dfa3f7783f82 in state ALARM triggered policy my-target-tracking-policy changing the desired capacity from 3 to 2. At 2020-02-11T18:34:31Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 2. At 2020-02-11T18:34:31Z instance i-06c4794c2499af1df was selected for termination.",
      "StartTime": "2020-02-11T18:34:31.268Z",
      "EndTime": "2020-02-11T18:34:53Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a\"...}",
      "AutoScalingGroupARN": "arn"
    },
    ...
  ]
}
```

Eine Beschreibung der Felder in der Ausgabe finden Sie unter [Aktivität](#) in der Amazon EC2 Auto Scaling API-Referenz.

Hilfe beim Abrufen der Skalierungsaktivitäten für eine gelöschte Gruppe und Informationen über die Arten von Fehlern, die auftreten können, und deren Behandlung finden Sie unter [Probleme in Amazon EC2 Auto Scaling beheben](#).

Eine Skalierungsrichtlinie für eine Auto-Scaling-Gruppe deaktivieren

In diesem Thema wird beschrieben, wie eine Skalierungsrichtlinie vorübergehend deaktiviert wird, damit keine Änderungen an der Anzahl der Instances in der Auto-Scaling-Gruppe ausgelöst werden. Wenn Sie eine Skalierungsrichtlinie deaktivieren, bleiben die Konfigurationsdetails erhalten, so dass Sie die Richtlinie schnell wieder aktivieren können. Dies ist einfacher, als eine Richtlinie vorübergehend zu löschen, wenn Sie sie nicht benötigen, und später neu zu erstellen.

Wenn eine Skalierungsrichtlinie deaktiviert ist, skaliert die Auto-Scaling-Gruppe nicht für die Metrikenalarms, die verletzt werden, während die Skalierungsrichtlinie deaktiviert ist. Alle noch laufenden Skalierungsaktivitäten werden jedoch nicht angehalten.

Beachten Sie, dass deaktivierte Skalierungsrichtlinien weiterhin für Ihre Kontingente für die Anzahl der Skalierungsrichtlinien zählen, die Sie einer Auto-Scaling-Gruppe hinzufügen können.

So deaktivieren Sie eine Skalierungsrichtlinie (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Aktivieren Sie auf der Registerkarte Automatische Skalierung unter Dynamische Skalierungsrichtlinien das Kontrollkästchen in der oberen rechten Ecke der gewünschten Skalierungsrichtlinie.
4. Scrollen Sie zum oberen Rand des Abschnitts Dynamische Skalierungsrichtlinien und wählen Sie Aktionen, Deaktivieren.

Wenn Sie bereit sind, die Skalierungsrichtlinie erneut zu aktivieren, wiederholen Sie diese Schritte, und wählen Sie dann Actions (Aktionen), Enable (Aktivieren). Nachdem Sie eine Skalierungsrichtlinie erneut aktiviert haben, kann Ihre Auto-Scaling-Gruppe sofort eine Skalierungsaktion initiieren, wenn derzeit Alarme im ALARM-Status vorhanden sind.

So deaktivieren Sie eine Skalierungsrichtlinie (AWS CLI):

Verwenden Sie den [put-scaling-policy](#) Befehl mit der `--no-enabled` Option wie folgt. Geben Sie alle Optionen in dem Befehl so an, wie Sie sie beim Erstellen der Richtlinie angeben würden.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
  --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
  --estimated-instance-warmup 360 \
  --target-tracking-configuration '{ "TargetValue": 70,
  "PredefinedMetricSpecification": { "PredefinedMetricType":
  "ASGAverageCPUUtilization" } }' \
  --no-enabled
```

So aktivieren Sie eine Skalierungsrichtlinie erneut (AWS CLI):

Verwenden Sie den [put-scaling-policy](#) Befehl mit der `--enabled` Option wie folgt. Geben Sie alle Optionen in dem Befehl so an, wie Sie sie beim Erstellen der Richtlinie angeben würden.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
  --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
  --estimated-instance-warmup 360 \
  --target-tracking-configuration '{ "TargetValue": 70,
  "PredefinedMetricSpecification": { "PredefinedMetricType":
  "ASGAverageCPUUtilization" } }' \
  --enabled
```

So beschreiben Sie eine Skalierungsrichtlinie (AWS CLI):

Verwenden Sie den [describe-policies](#) Befehl, um den aktivierten Status einer Skalierungsrichtlinie zu überprüfen.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg \
  --policy-names my-scaling-policy
```

Es folgt eine Beispielausgabe.

```
{
  "ScalingPolicies": [
    {
      "AutoScalingGroupName": "my-asg",
      "PolicyName": "my-scaling-policy",
      "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:1d52783a-b03b-4710-
bb0e-549fd64378cc:autoScalingGroupName/my-asg:policyName/my-scaling-policy",
```

```
    "PolicyType": "TargetTrackingScaling",
    "StepAdjustments": [],
    "Alarms": [
      {
        "AlarmName": "TargetTracking-my-asg-
AlarmHigh-9ca53fdd-7cf5-4223-938a-ae1199204502",
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-
ae1199204502"
      },
      {
        "AlarmName": "TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c",
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c"
      }
    ],
    "TargetTrackingConfiguration": {
      "PredefinedMetricSpecification": {
        "PredefinedMetricType": "ASGAverageCPUUtilization"
      },
      "TargetValue": 70.0,
      "DisableScaleIn": false
    },
    "Enabled": true
  }
]
```

Löschen Sie eine Skalierungsrichtlinie für eine Auto Scaling Scaling-Gruppe

Wenn Sie für die Skalierung keine Richtlinie mehr benötigen, können Sie diese löschen. Je nach Art der Skalierungsrichtlinie müssen Sie möglicherweise auch die CloudWatch Alarmer löschen. Durch das Löschen einer Skalierungsrichtlinie für die Zielverfolgung werden auch alle zugehörigen CloudWatch Alarmer gelöscht. Durch das Löschen einer schrittweisen Skalierungsrichtlinie oder einer einfachen Skalierungsrichtlinie wird die zugrunde liegende Alarmaktion gelöscht, der CloudWatch Alarm wird jedoch nicht gelöscht, auch wenn ihm keine Aktion mehr zugeordnet ist.

Löschen einer Skalierungsrichtlinie (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.
3. Aktivieren Sie auf der Registerkarte Automatische Skalierung unter Dynamische Skalierungsrichtlinien das Kontrollkästchen in der oberen rechten Ecke der gewünschten Skalierungsrichtlinie.
4. Scrollen Sie zum oberen Rand des Abschnitts Dynamische Skalierungsrichtlinien und wählen Sie Aktionen, Löschen.
5. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Ja, löschen.
6. (Optional) Wenn Sie eine Step Scaling-Richtlinie oder eine einfache Skalierungsrichtlinie gelöscht haben, gehen Sie wie folgt vor, um den CloudWatch Alarm zu löschen, der mit der Richtlinie verknüpft war. Sie können diese Teilschritte überspringen, um den Alarm für die zukünftige Verwendung beizubehalten.
 - a. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
 - b. Wählen Sie im Navigationsbereich Alarms (Alarme) aus.
 - c. Wählen Sie zunächst den Alarm (z. B. Step-Scaling-AlarmHigh-AddCapacity) und anschließend Action (Aktion), Delete (Löschen) aus.
 - d. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Delete (Löschen).

So rufen Sie die Skalierungsrichtlinien für eine Auto-Scaling-Gruppe ab (AWS CLI)

Bevor Sie eine Skalierungsrichtlinie löschen, verwenden Sie den Befehl [describe-policies](#), um die Skalierungsrichtlinien zu ermitteln, die für die Auto-Scaling-Gruppe erstellt wurden. Sie können die Ausgabe beim Löschen der Richtlinie und der CloudWatch-Alarme verwenden.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
```

Sie können die Ergebnisse mithilfe des Parameters `--query` nach dem Typ der Skalierungsrichtlinie filtern. Diese Syntax für `query` funktioniert unter Linux oder macOS. Unter Windows ändern Sie die einfachen Anführungszeichen in doppelte Anführungszeichen.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
```

```
--query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

Es folgt eine Beispielausgabe.

```
[
  {
    "AutoScalingGroupName": "my-asg",
    "PolicyName": "cpu50-target-tracking-scaling-policy",
    "PolicyARN": "PolicyARN",
    "PolicyType": "TargetTrackingScaling",
    "StepAdjustments": [],
    "Alarms": [
      {
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
        "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
      },
      {
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
        "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
      }
    ],
    "TargetTrackingConfiguration": {
      "PredefinedMetricSpecification": {
        "PredefinedMetricType": "ASGAverageCPUUtilization"
      },
      "TargetValue": 50.0,
      "DisableScaleIn": false
    },
    "Enabled": true
  }
]
```

So löschen Sie Ihre Skalierungsrichtlinie (AWS CLI)

Verwenden Sie den folgenden Befehl [delete-policy](#).

```
aws autoscaling delete-policy --auto-scaling-group-name my-asg \
```

```
--policy-name cpu50-target-tracking-scaling-policy
```

Um deinen CloudWatch Alarm zu löschen (AWS CLI)

Verwenden Sie für schrittweise und einfache Skalierungsrichtlinien den Befehl [delete-alarms](#), um den CloudWatch Alarm zu löschen, der der Richtlinie zugeordnet war. Sie können diesen Schritt überspringen, um den Alarm für die spätere Verwendung beizubehalten. Sie können einen oder mehrere Alarme gleichzeitig löschen. Sie können beispielsweise den folgenden Befehl verwenden, um die Alarme Step-Scaling-AlarmHigh-AddCapacity und Step-Scaling-AlarmLow-RemoveCapacity zu löschen.

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-RemoveCapacity
```

Beispiel für Skalierungsrichtlinien für AWS CLI

Sie können Skalierungsrichtlinien für Amazon EC2 Auto Scaling über AWS-Managementkonsole, AWS Command Line Interface (AWS CLI) oder SDKs erstellen.

Die folgenden Beispiele zeigen, wie Sie mit dem AWS CLI [put-scaling-policy](#) Befehl Skalierungsrichtlinien für Amazon EC2 Auto Scaling erstellen können. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

Informationen zu den ersten Schritten beim Schreiben von Skalierungsrichtlinien mithilfe von finden Sie in den einführenden Übungen unter [Skalierungsrichtlinien für die Ziel-Nachverfolgung](#) und [Schrittweise und einfache Skalierungsrichtlinien](#). AWS CLI

Beispiel 1: So wenden Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung mit einer vordefinierten Metrikspezifikation an

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \  
  --target-tracking-configuration file://config.json  
{  
  "TargetValue": 50.0,  
  "PredefinedMetricSpecification": {  
    "PredefinedMetricType": "ASGAverageCPUUtilization"  
  }  
}
```

Weitere Informationen finden Sie [PredefinedMetricSpecification](#) in der Amazon EC2 Auto Scaling API-Referenz.

Note

Wenn sich die Datei nicht im aktuellen Verzeichnis befindet, geben Sie den vollständigen Dateipfad ein. Weitere Informationen zum Lesen von AWS CLI Parameterwerten aus einer Datei finden Sie unter [Laden von AWS CLI Parametern aus einer Datei](#) im AWS Command Line Interface Benutzerhandbuch.

Beispiel 2: So wenden Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung mit einer benutzerdefinierten Metrikspezifikation an

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
    "MetricName": "MyBacklogPerInstance",
    "Namespace": "MyNamespace",
    "Dimensions": [{
      "Name": "MyOptionalMetricDimensionName",
      "Value": "MyOptionalMetricDimensionValue"
    }],
    "Statistic": "Average",
    "Unit": "None"
  }
}
```

Weitere Informationen finden Sie [CustomizedMetricSpecification](#) in der Amazon EC2 Auto Scaling API-Referenz.

Beispiel 3: So wenden Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung nur für die horizontale Skalierung nach oben an

```
aws autoscaling put-scaling-policy --policy-name alb1000-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
```

```
--target-tracking-configuration file://config.json
{
  "TargetValue": 1000.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ALBRequestCountPerTarget",
    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-
group/943f017f100becff"
  },
  "DisableScaleIn": true
}
```

Beispiel 4: So wenden Sie eine Schrittskalierungsrichtlinie für die horizontale Skalierung nach oben an

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
MetricIntervalLowerBound=10.0,MetricIntervalUpperBound=20.0,ScalingAdjustment=10 \
MetricIntervalLowerBound=20.0,MetricIntervalUpperBound=30.0,ScalingAdjustment=20 \
  MetricIntervalLowerBound=30.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Richtlinie. Sie benötigen den ARN, wenn Sie den CloudWatch Alarm erstellen.

Beispiel 5: So wenden Sie eine Schrittskalierungsrichtlinie für die horizontale Skalierung nach unten an

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Richtlinie. Sie benötigen den ARN, wenn Sie den CloudWatch Alarm erstellen.

Beispiel 6: So wenden Sie eine einfache Skalierungsrichtlinie für die horizontale Skalierung nach oben an

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \  
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \  
  --adjustment-type PercentChangeInCapacity --min-adjustment-magnitude 2
```

Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Richtlinie. Sie benötigen den ARN, wenn Sie den CloudWatch Alarm erstellen.

Beispiel 7: So wenden Sie eine einfache Skalierungsrichtlinie für die horizontale Skalierung nach unten an

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \  
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \  
  --adjustment-type ChangeInCapacity --cooldown 180
```

Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Richtlinie. Sie benötigen den ARN, wenn Sie den CloudWatch Alarm erstellen.

Prädiktive Skalierung für Amazon EC2 Auto Scaling

Bei der prädiktiven Skalierung werden historische Lastdaten analysiert, um tägliche oder wöchentliche Muster im Verkehrsfluss zu erkennen. Es verwendet diese Informationen, um future Kapazitätsanforderungen zu prognostizieren, sodass Amazon EC2 Auto Scaling die Kapazität Ihrer Auto Scaling-Gruppe proaktiv erhöhen kann, um sie an die erwartete Auslastung anzupassen.

Die prädiktive Skalierung eignet sich gut für Situationen, in denen Sie Folgendes haben:

- Zyklischen Datenverkehr, z. B. hohe Auslastung von Ressourcen während der normalen Geschäftszeiten und niedrige Auslastung von Ressourcen am Abend und am Wochenende
- Wiederkehrende on-and-off Workload-Muster, wie z. B. Stapelverarbeitung, Tests oder regelmäßige Datenanalysen
- Anwendungen, die eine lange Zeit in Anspruch nehmen, was zu einer spürbaren Latenzauswirkung auf die Anwendungsleistung bei Aufskalierungsereignissen führt

Im Allgemeinen sollten Sie prädiktive Skalierung in Betracht ziehen, wenn Sie regelmäßige Datenverkehrszuwächse haben und Anwendungen nutzen, deren Initialisierung lange dauert. Mit

der prädiktiven Skalierung können Sie im Vergleich mit nur dynamischer Skalierung (die reaktiv ist) schneller skalieren, indem Sie die Kapazität vor der prognostizierten Last starten. Durch vorausschauende Skalierung können Sie möglicherweise auch Geld bei Ihrer EC2-Rechnung sparen, da Sie vermeiden, dass Sie zu viel Kapazität bereitstellen müssen.

Nehmen wir als Beispiel eine Anwendung, die eine hohe Auslastung während der Geschäftszeiten und eine geringe Nutzung über Nacht aufweist. Zu Beginn eines jeden Werktages kann die prädiktive Skalierung die Kapazität vor dem ersten Zustrom des Datenverkehrs erhöhen. Auf diese Weise kann Ihre Anwendung eine hohe Verfügbarkeit und Leistung aufrechterhalten, wenn sie von einer geringeren Auslastung zu einem Zeitraum mit höherer Auslastung übergeht. Sie müssen nicht warten, bis die dynamische Skalierung auf sich ändernden Datenverkehr reagiert. Sie müssen auch keine Zeit damit verbringen, die Lastmuster Ihrer Anwendung zu überprüfen und mithilfe der geplanten Skalierung die richtige Kapazität zu planen.

Topics

- [So funktioniert Auto Scaling](#)
- [Erstellen Sie eine Richtlinie für vorausschauende Skalierung für eine Auto Scaling Scaling-Gruppe](#)
- [Auswertung Ihrer Richtlinien für prädiktive Skalierung](#)
- [Überschreiben von Prognosewerten mithilfe geplanter Aktionen](#)
- [Erweiterte prädiktive Skalierungsrichtlinie unter Verwendung benutzerdefinierter Metriken](#)

So funktioniert Auto Scaling

In diesem Thema wird erklärt, wie Predictive Scaling funktioniert, und es wird beschrieben, was bei der Erstellung einer Predictive Scaling-Richtlinie zu beachten ist.

Themen

- [Funktionsweise](#)
- [Maximales Kapazitätslimit](#)
- [Überlegungen](#)
- [Unterstützte Regionen](#)

Funktionsweise

Um Predictive Scaling zu verwenden, erstellen Sie eine Predictive Scaling-Richtlinie, die die zu überwachende und zu analysierende CloudWatch Metrik festlegt. Damit die prädiktive Skalierung

mit der Prognose future Werte beginnen kann, muss diese Metrik Daten für mindestens 24 Stunden enthalten.

Nachdem Sie die Richtlinie erstellt haben, beginnt die prädiktive Skalierung mit der Analyse von Metrikdaten der letzten 14 Tage, um Muster zu identifizieren. Anhand dieser Analyse wird eine stündliche Prognose des Kapazitätsbedarfs für die nächsten 48 Stunden erstellt. Die Prognose wird alle 6 Stunden anhand der neuesten CloudWatch Daten aktualisiert. Wenn neue Daten eintreffen, kann die prädiktive Skalierung die Genauigkeit zukünftiger Prognosen kontinuierlich verbessern.

Wenn Sie die prädiktive Skalierung zum ersten Mal aktivieren, wird sie im Modus nur Prognose ausgeführt. In diesem Modus werden Kapazitätsprognosen generiert, Ihre Auto Scaling-Gruppe jedoch nicht auf der Grundlage dieser Prognosen skaliert. Auf diese Weise können Sie die Genauigkeit und Eignung der Prognose bewerten. Sie können Prognosedaten mithilfe der `GetPredictiveScalingForecast` API-Operation oder der AWS-Managementkonsole anzeigen.

Nachdem Sie die Prognosedaten überprüft und beschlossen haben, mit der Skalierung auf der Grundlage dieser Daten zu beginnen, schalten Sie die Skalierungsrichtlinie in den Prognose- und Skalierungsmodus um. In diesem Modus:

- Wenn in der Prognose ein Anstieg der Auslastung erwartet wird, wird Amazon EC2 Auto Scaling die Kapazität durch Skalierung erhöhen.
- Wenn in der Prognose ein Rückgang der Auslastung erwartet wird, wird sie nicht skaliert, um Kapazität zu verringern. Wenn Sie nicht mehr benötigte Kapazität entfernen möchten, müssen Sie dynamische Skalierungsrichtlinien erstellen.

Standardmäßig skaliert Amazon EC2 Auto Scaling Ihre Auto Scaling-Gruppe zu Beginn jeder Stunde auf der Grundlage der Prognose für diese Stunde. Sie können optional eine frühere Startzeit angeben, indem Sie die `SchedulingBufferTime` Eigenschaft im `PutScalingPolicy` API-Vorgang oder die Einstellung `Pre-Launch Instances` in der verwenden. AWS-Managementkonsole Dies veranlasst Amazon EC2 Auto Scaling dazu, neue Instances vor dem prognostizierten Bedarf zu starten, sodass sie Zeit haben, zu starten und für die Verarbeitung des Datenverkehrs bereit zu sein.

Um das Starten neuer Instances vor dem prognostizierten Bedarf zu unterstützen, empfehlen wir Ihnen dringend, das Standard-Instance-Warmup für Ihre Auto Scaling Scaling-Gruppe zu aktivieren. Dies gibt einen Zeitraum nach einer Scale-Out-Aktivität an, in dem Amazon EC2 Auto Scaling nicht skaliert, auch wenn dynamische Skalierungsrichtlinien darauf hinweisen, dass die Kapazität verringert werden sollte. Auf diese Weise können Sie sicherstellen, dass neu gestartete Instances ausreichend Zeit haben, um mit der Bearbeitung des erhöhten Datenverkehrs zu beginnen, bevor sie für Scale-

In-Operationen in Betracht gezogen werden. Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

Maximales Kapazitätslimit

Auto Scaling Scaling-Gruppen haben eine maximale Kapazitätseinstellung, die die maximale Anzahl von EC2-Instances begrenzt, die für die Gruppe gestartet werden können. Wenn Skalierungsrichtlinien festgelegt sind, können sie die Kapazität standardmäßig nicht über die maximale Kapazität hinaus erhöhen.

Alternativ können Sie zulassen, dass die maximale Kapazität der Gruppe automatisch erhöht wird, wenn sich die prognostizierte Kapazität der Auto Scaling-Gruppe der maximalen Kapazität der Auto Scaling Scaling-Gruppe nähert oder diese überschreitet. Um dieses Verhalten zu aktivieren, verwenden Sie die `MaxCapacityBuffer` Eigenschaften `MaxCapacityBreachBehavior` und im `PutScalingPolicy` API-Vorgang oder die Einstellung für das Verhalten „Max. Kapazität“ in AWS-Managementkonsole.

Warning

Seien Sie vorsichtig, wenn Sie zulassen, dass die maximale Kapazität automatisch erhöht wird. Dies kann dazu führen, dass mehr Instances gestartet werden als beabsichtigt, wenn die erhöhte maximale Kapazität nicht überwacht und verwaltet wird. Die erhöhte maximale Kapazität wird dann zur neuen normalen maximalen Kapazität für die Auto Scaling Scaling-Gruppe, bis Sie sie manuell aktualisieren. Die maximale Kapazität wird nicht automatisch wieder auf das ursprüngliche Maximum reduziert.

Überlegungen

- Bestätigen Sie, ob die prädiktive Skalierung für Ihren Workload geeignet ist. Ein Workload eignet sich gut für die prädiktive Skalierung, wenn er wiederkehrende Lastmuster aufweist, die spezifisch für den Wochentag oder die Tageszeit sind. Um dies zu überprüfen, konfigurieren Sie die Richtlinien für prädiktive Skalierung im Modus Nur Prognose und lesen dann die Empfehlungen in der Konsole. Amazon EC2 Auto Scaling bietet Empfehlungen basierend auf Beobachtungen zur potenziellen Richtlinienleistung. Bewerten Sie die Prognose und die Empfehlungen, bevor Sie Ihre Anwendung durch prädiktive Skalierung aktiv skalieren lassen.
- Für die prädiktive Skalierung werden mindestens 24 Stunden an historischen Daten benötigt, damit mit der Prognose begonnen werden kann. Prognosen sind jedoch effektiver, wenn Verlaufsdaten

für zwei volle Wochen zur Verfügung stehen. Wenn Sie Ihre Anwendung aktualisieren, indem Sie eine neue Auto-Scaling-Gruppe erstellen und die alte löschen, benötigt die neue Auto-Scaling-Gruppe 24 Stunden an historischen Lastdaten, bevor die prädiktive Skalierung wieder Prognosen generieren kann. Sie können benutzerdefinierte Metriken verwenden, um Metriken über alte und neue Auto-Scaling-Gruppen hinweg zu aggregieren. Andernfalls müssen Sie möglicherweise einige Tage auf eine genauere Prognose warten.

- Wählen Sie eine Lastmetrik, die die volle Auslastung Ihrer Anwendung genau wiedergibt und den Aspekt Ihrer Anwendung darstellt, der für die Skalierung am wichtigsten ist.
- Die dynamische Skalierung mit vorausschauender Skalierung hilft Ihnen dabei, die Nachfragekurve für Ihre Anwendung genau zu verfolgen, indem Sie in Zeiten mit geringem Datenverkehr nach oben skalieren und bei höherem Datenverkehr wieder herunterskalieren. Wenn mehrere Skalierungsrichtlinien aktiv sind, bestimmt jede Richtlinie die gewünschte Kapazität unabhängig, und die gewünschte Kapazität wird auf das Maximum dieser Richtlinien festgelegt. Wenn beispielsweise 10 Instances an der Zielauslastung in einer Skalierungsrichtlinie für Zielverfolgung verbleiben müssen und acht Instances in einer Richtlinie zur prädiktiven Skalierung an der Zielauslastung bleiben müssen, wird die gewünschte Kapazität der Gruppe auf zehn festgelegt. Wenn Sie mit dynamischer Skalierung noch nicht vertraut sind, empfehlen wir die Verwendung von Skalierungsrichtlinien für die Zielverfolgung. Weitere Informationen finden Sie unter [Dynamische Skalierung für Amazon EC2 Auto Scaling](#).
- Eine zentrale Annahme der vorausschauenden Skalierung ist, dass die Auto-Scaling-Gruppe homogen ist und alle Instances von gleicher Kapazität sind. Wenn dies für Ihre Gruppe nicht zutrifft, kann die prognostizierte Kapazität ungenau sein. Seien Sie daher vorsichtig, wenn Sie Richtlinien zur vorausschauenden Skalierung für [Gruppen mit gemischten Instanzen](#) erstellen, da Instances verschiedener Typen mit ungleicher Kapazität bereitgestellt werden können. Im Folgenden finden Sie einige Beispiele, bei denen die prognostizierte Kapazität ungenau sein wird:
 - Ihre Richtlinie zur vorausschauenden Skalierung basiert auf der CPU-Auslastung, aber die Anzahl von v CPUs auf jeder Auto Scaling Scaling-Instance variiert je nach Instance-Typ.
 - Ihre Richtlinie zur vorausschauenden Skalierung basiert auf einem Netzwerk-In- oder Netzwerkausgang, aber der Netzwerkbandbreitendurchsatz für jede Auto-Scaling-Instance variiert je nach Instance-Typen. Zum Beispiel ähneln sich die Instance-Typen M5 und M5n, der M5n-Instance-Typ liefert jedoch einen deutlich höheren Netzwerkdurchsatz.

Unterstützte Regionen

- USA Ost (Nord-Virginia)

- USA Ost (Ohio)
- USA West (Nordkalifornien)
- USA West (Oregon)
- Africa (Cape Town)
- Asien-Pazifik (Hongkong)
- Asien-Pazifik (Hyderabad)
- Asien-Pazifik (Jakarta)
- Asien-Pazifik (Melbourne)
- Asien-Pazifik (Mumbai)
- Asien-Pazifik (Osaka)
- Asien-Pazifik (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Asien-Pazifik (Tokio)
- Canada (Central)
- Kanada West (Calgary)
- China (Peking)
- China (Ningxia)
- Europe (Frankfurt)
- Europa (Irland)
- Europa (London)
- Europa (Milan)
- Europa (Paris)
- Europa (Spain)
- Europa (Stockholm)
- Europa (Zürich)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- Naher Osten (VAE)
- Südamerika (São Paulo)

- AWS GovCloud (USA Ost)
- AWS GovCloud (US-West)

Erstellen Sie eine Richtlinie für vorausschauende Skalierung für eine Auto Scaling Scaling-Gruppe

Die folgenden Verfahren helfen Ihnen bei der Erstellung einer Richtlinie für vorausschauende Skalierung mithilfe von oder. AWS-Managementkonsole AWS CLI

Wenn die Auto-Scaling-Gruppe neu ist, muss es mindestens 24 Stunden an Daten liefern, bevor Amazon EC2 Auto Scaling eine Prognose erstellen kann.

Inhalt

- [Erstellen einer Richtlinie für die prädiktive Skalierung \(Konsole\)](#)
- [Erstellen einer Richtlinie für die prädiktive Skalierung \(AWS CLI\)](#)

Erstellen einer Richtlinie für die prädiktive Skalierung (Konsole)

Wenn Sie zum ersten Mal eine Richtlinie für vorausschauende Skalierung erstellen, empfehlen wir, die Konsole zu verwenden, um mehrere Richtlinien für die prädiktive Skalierung im Modus „Nur Prognose“ zu erstellen. Auf diese Weise können Sie die potenziellen Auswirkungen verschiedener Metriken und Zielwerte testen. Sie können mehrere Richtlinien für die prädiktive Skalierung für jede Auto-Scaling-Gruppe erstellen, aber nur eine der Richtlinien kann für die aktive Skalierung verwendet werden.

Erstellen einer Richtlinie für die prädiktive Skalierung (vordefinierte Metriken)

Führen Sie das folgende Verfahren aus, um eine Richtlinie für die prädiktive Skalierung unter Verwendung vordefinierter Metriken (CPU, Netzwerk-I/O oder Anzahl der Anfragen an den Application Load Balancer) zu erstellen. Der einfachste Weg, eine Richtlinie für die prädiktive Skalierung zu erstellen, besteht darin, vordefinierte Metriken zu verwenden. Wenn Sie stattdessen benutzerdefinierte Metriken verwenden möchten, siehe [Erstellen einer Richtlinie für die prädiktive Skalierung in der Konsole \(benutzerdefinierte Metriken\)](#).

So erstellen Sie eine Richtlinie für die prädiktive Skalierung

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.

2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Automatic scaling (Automatische Skalierung) unter Scaling policies (Skalierungsrichtlinien) die Option Create predictive scaling policy (Richtlinie für die prädiktive Skalierung erstellen) aus.
4. Geben Sie einen Namen für die Richtlinie ein.
5. Aktivieren Sie Skalierung basierend auf Prognose, um Amazon EC2 Auto Scaling die Berechtigung zu erteilen, sofort mit der Skalierung zu beginnen.

Um die Richtlinie im Modus Nur Prognose zu belassen, bleibt Skalierung basierend auf Prognose deaktiviert.

6. Für Metriken wählen Sie Ihre Metriken aus der Liste der Optionen aus. Zu den Optionen gehören CPU, Netzwerkeingang, Netzwerkausgang, Anzahl der Application Load Balancer und Benutzerdefiniertes Metrikpaar.

Wenn Sie Anzahl der Application Load Balancer pro Ziel auswählen, wählen Sie anschließend in Zielgruppe eine Zielgruppe aus. Anzahl der Application Load Balancer pro Ziel wird nur unterstützt, wenn Sie eine Application Load Balancer-Zielgruppe an Ihre Auto-Scaling-Gruppe angehängt haben.

Wenn Sie Benutzerdefiniertes Metrikpaar auswählen, wählen Sie dann aus individuelle Metriken aus den Dropdown-Listen für Lastmetrik und Skalierungsmetrik aus.

7. Für Zielauslastung geben Sie den Zielwert ein, den Amazon EC2 Auto Scaling beibehalten werden soll. Amazon EC2 Auto Scaling skaliert Ihre Kapazität, bis die durchschnittliche Auslastung der Zielauslastung entspricht oder bis sie die von Ihnen angegebene maximale Anzahl von Instances erreicht.

Wenn Ihre Skalierungsmetrik ...	Dann ist die Zielauslastung ...
CPU	Der prozentuale CPU-Anteil, den jede Instance idealerweise verwenden sollte.
Netzwerkeingang	Die durchschnittliche Anzahl von Bytes pro Minute, die jede Instance idealerweise empfangen sollte.

Wenn Ihre Skalierungsmetrik ...	Dann ist die Zielauslastung ...
Netzwerkausgang	Die durchschnittliche Anzahl von Bytes pro Minute, die jede Instance idealerweise senden sollte.
Anzahl der Application Load Balancer pro Ziel	Die durchschnittliche Anzahl von Anfragen pro Minute, die jede Instance idealerweise empfangen sollte.

8. (Optional) Für Vorabstarten von Instances wählen Sie aus, wie weit im Voraus Ihre Instances gestartet werden sollen, bevor die Prognose die Last erhöht.
9. (Optional) Für Verhalten bei max. Kapazität wählen Sie aus, ob Sie Amazon EC2 Auto Scaling höher als die maximale Kapazität der Gruppe skalieren lassen, wenn die prognostizierte Kapazität das definierte Maximum überschreitet. Wenn Sie diese Einstellung aktivieren, kann die Skalierung in Zeiten erfolgen, in denen der höchste Datenverkehr vorausgesagt wird.
10. (Optional) Für Maximale Pufferkapazität oberhalb der prognostizierten Kapazität wählen Sie die zusätzliche Kapazität aus, die Sie verwenden möchten, wenn die prognostizierte Kapazität bei der maximalen Kapazität liegt oder diese überschreitet. Der Wert wird als Prozentsatz relativ zur prognostizierten Kapazität angegeben. Beispiel: Wenn der Puffer 10 ist, bedeutet dies, dass ein Puffer von 10 Prozent vorhanden ist. Wenn daher die prognostizierte Kapazität 50 ist und die maximale Kapazität 40 ist, dann ist die effektive maximale Kapazität 55.

Wenn die Option 0 ist, kann Amazon EC2 Auto Scaling eine Kapazität skalieren, die höher als die maximale Kapazität ist, um der prognostizierten Kapazität zu entsprechen, diese aber nicht zu überschreiten.


11. Klicken Sie auf Erstellen einer Richtlinie für die prädiktive Skalierung.

Erstellen einer Richtlinie für die prädiktive Skalierung in der Konsole (benutzerdefinierte Metriken)

Führen Sie das folgende Verfahren aus, um eine Richtlinie für die prädiktive Skalierung unter Verwendung benutzerdefinierter Metriken zu erstellen. Benutzerdefinierte Metriken können andere Metriken enthalten, die von Ihnen bereitgestellt werden, CloudWatch oder Metriken, in denen Sie CloudWatch veröffentlichen. Informationen zur Verwendung der CPU-, Netzwerk-I/O- oder Application Load Balancer Balancer-Anforderungsanzahl pro Ziel finden Sie unter [Erstellen einer Richtlinie für die prädiktive Skalierung \(vordefinierte Metriken\)](#).

Zur Erstellung einer Richtlinie für die prädiktive Skalierung unter Verwendung benutzerdefinierter Metriken:

- Sie müssen die Rohabfragen angeben, die es Amazon EC2 Auto Scaling ermöglichen, mit den Metriken in CloudWatch zu interagieren. Weitere Informationen finden Sie unter [Erweiterte prädiktive Skalierungsrichtlinie unter Verwendung benutzerdefinierter Metriken](#). Um sicherzustellen, dass Amazon EC2 Auto Scaling die Metrikdaten aus extrahieren kann CloudWatch, stellen Sie sicher, dass jede Abfrage Datenpunkte zurückgibt. Bestätigen Sie dies mithilfe der CloudWatch Konsole oder der CloudWatch [GetMetricData](#)API-Operation.

 Note

Wir stellen Beispiel-JSON-Nutzlasten im JSON-Editor in der Amazon EC2 Auto Scaling Konsole zur Verfügung. Diese Beispiele geben Ihnen eine Referenz für die Schlüssel-Wert-Paare, die erforderlich sind, um andere CloudWatch Metriken hinzuzufügen, die von bereitgestellt werden AWS oder für die Sie zuvor veröffentlicht haben. CloudWatch Sie können sie als Ausgangspunkt verwenden und sie dann an Ihre Bedürfnisse anpassen.

- Wenn Sie metrische Berechnungen verwenden, müssen Sie den JSON manuell so erstellen, dass er zu Ihrem individuellen Szenario passt. Weitere Informationen finden Sie unter [Metrikberechnungs-Ausdrücke verwenden](#). Bevor Sie metrische Berechnungen in Ihrer Richtlinie verwenden, stellen Sie sicher, dass Metrikabfragen, die auf metrischen mathematischen Ausdrücken basieren, gültig sind, und geben Sie eine einzelne Zeitreihe zurück. Bestätigen Sie dies mithilfe der CloudWatch Konsole oder der CloudWatch [GetMetricData](#)API-Operation.

Wenn Sie in einer Abfrage einen Fehler machen, indem Sie falsche Daten angeben, z. B. den falschen Auto-Scaling-Gruppennamen, enthält die Prognose keine Daten. Informationen zur Behebung von Problemen mit benutzerdefinierten Metriken finden Sie unter [Überlegungen zu benutzerdefinierten Metriken in einer Richtlinie zur prädiktiven Skalierung](#).

So erstellen Sie eine Richtlinie für die prädiktive Skalierung

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Automatic scaling (Automatische Skalierung) unter Scaling policies (Skalierungsrichtlinien) die Option Create predictive scaling policy (Richtlinie für die prädiktive Skalierung erstellen) aus.

4. Geben Sie einen Namen für die Richtlinie ein.
5. Aktivieren Sie Skalierung basierend auf Prognose, um Amazon EC2 Auto Scaling die Berechtigung zu erteilen, sofort mit der Skalierung zu beginnen.

Um die Richtlinie im Modus Nur Prognose zu belassen, bleibt Skalierung basierend auf Prognose deaktiviert.

6. Wählen Sie für Metriken die Option Benutzerdefiniertes Metrikpaar aus.
 - a. Wählen Sie unter Metrik laden die Option Benutzerdefinierte CloudWatch Metrik aus, um eine benutzerdefinierte Metrik zu verwenden. Erstellen Sie die JSON-Nutzlast, die die Metrik der Richtlinie enthält, und fügen Sie diese in das JSON-Editorfeld ein. Ersetzen Sie dabei den Inhalt des Feldes.
 - b. Wählen Sie für Skalierungsmetrik die Option Benutzerdefinierte CloudWatch Metrik aus, um eine benutzerdefinierte Metrik zu verwenden. Erstellen Sie die JSON-Nutzlast, die die Definition der Skalierungsmetrik für die Richtlinie enthält, und fügen Sie sie in das JSON-Editorfeld ein. Ersetzen Sie dabei den Inhalt des Feldes.
 - c. (Optional) Um eine benutzerdefinierte Kapazitätsmetrik hinzuzufügen, aktivieren Sie das Kontrollkästchen Benutzerdefinierte Kapazitätsmetrik hinzufügen. Erstellen Sie die JSON-Nutzlast, die die Definition der Kapazitätsmetrik für die Richtlinie enthält, und fügen Sie sie in das JSON-Editorfeld ein. Ersetzen Sie dabei den Inhalt des Feldes.

Sie müssen diese Option nur aktivieren, um eine neue Zeitreihe für Kapazität zu erstellen, wenn sich Ihre Kapazitätsmetrikdaten über mehrere Auto-Scaling-Gruppen erstrecken. In diesem Fall müssen Sie metrische Mathematik verwenden, um die Daten zu einer einzigen Zeitreihe zu aggregieren.

7. Für Zielauslastung geben Sie den Zielwert ein, den Amazon EC2 Auto Scaling beibehalten werden soll. Amazon EC2 Auto Scaling skaliert Ihre Kapazität, bis die durchschnittliche Auslastung der Zielauslastung entspricht oder bis sie die von Ihnen angegebene maximale Anzahl von Instances erreicht.
8. (Optional) Für Vorabstarten von Instances wählen Sie aus, wie weit im Voraus Ihre Instances gestartet werden sollen, bevor die Prognose die Last erhöht.
9. (Optional) Für Verhalten bei max. Kapazität wählen Sie aus, ob Sie Amazon EC2 Auto Scaling höher als die maximale Kapazität der Gruppe skalieren lassen, wenn die prognostizierte Kapazität das definierte Maximum überschreitet. Wenn Sie diese Einstellung aktivieren, kann die Skalierung in Zeiten erfolgen, in denen der höchste Datenverkehr vorausgesagt wird.

10. (Optional) Für Maximale Pufferkapazität oberhalb der prognostizierten Kapazität wählen Sie die zusätzliche Kapazität aus, die Sie verwenden möchten, wenn die prognostizierte Kapazität bei der maximalen Kapazität liegt oder diese überschreitet. Der Wert wird als Prozentsatz relativ zur prognostizierten Kapazität angegeben. Beispiel: Wenn der Puffer 10 ist, bedeutet dies, dass ein Puffer von 10 Prozent vorhanden ist. Wenn daher die prognostizierte Kapazität 50 ist und die maximale Kapazität 40 ist, dann ist die effektive maximale Kapazität 55.

Wenn die Option 0 ist, kann Amazon EC2 Auto Scaling eine Kapazität skalieren, die höher als die maximale Kapazität ist, um der prognostizierten Kapazität zu entsprechen, diese aber nicht zu überschreiten.

11. Klicken Sie auf Erstellen einer Richtlinie für die prädiktive Skalierung.

Erstellen einer Richtlinie für die prädiktive Skalierung (AWS CLI)

Gehen Sie AWS CLI wie folgt vor, um Predictive Scaling-Richtlinien für Ihre Auto Scaling Scaling-Gruppe zu konfigurieren. Ersetzen Sie jeden *user input placeholder* durch Ihre Informationen.

Weitere Informationen zu den CloudWatch Metriken, die Sie angeben können, finden Sie [PredictiveScalingMetricSpecification](#) in der Amazon EC2 Auto Scaling API-Referenz.

Beispiel 1: Eine Richtlinie für die prädiktive Skalierung, die Prognosen erstellt, aber nicht skaliert

Die folgende Beispielrichtlinie zeigt eine vollständige Richtlinienkonfiguration, die CPU-Auslastungsmetriken für die prädiktive Skalierung mit einer Zielauslastung von 40 verwendet. ForecastOnly wird standardmäßig verwendet, es sei denn, Sie geben explizit an, welcher Modus verwendet werden soll. Speichern Sie diese Konfiguration in einer Datei mit dem Namen config.json.

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 40,
      "PredefinedMetricPairSpecification": {
        "PredefinedMetricType": "ASGCPUtilization"
      }
    }
  ]
}
```

Um die Richtlinie von der Befehlszeile aus zu erstellen, führen Sie den [put-scaling-policy](#) Befehl mit der angegebenen Konfigurationsdatei aus, wie im folgenden Beispiel gezeigt.

```
aws autoscaling put-scaling-policy --policy-name cpu40-predictive-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \  
  --predictive-scaling-configuration file://config.json
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er den Amazon-Ressourcennamen (ARN) der Richtlinie zurück.

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-  
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/cpu40-predictive-scaling-  
policy",  
  "Alarms": []  
}
```

Beispiel 2: Eine Richtlinie für die prädiktive Skalierung, die Prognosen erstellt und skaliert

Fügen Sie für eine Richtlinie, mit der Amazon EC2 Auto Scaling prognostiziert und skaliert werden kann, die Eigenschaft `Mode` mit einem Wert von `ForecastAndScale` hinzu. Das folgende Beispiel zeigt eine Richtlinienkonfiguration, die Anforderungsanzahlmetriken der Application Load Balancer verwendet. Die Zielauslastung ist `1000` und die prädiktive Skalierung ist auf den Modus `ForecastAndScale` eingestellt.

```
{  
  "MetricSpecifications": [  
    {  
      "TargetValue": 1000,  
      "PredefinedMetricPairSpecification": {  
        "PredefinedMetricType": "ALBRequestCount",  
        "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-  
target-group/943f017f100becff"  
      }  
    }  
  ],  
  "Mode": "ForecastAndScale"  
}
```

Um diese Richtlinie zu erstellen, führen Sie den [put-scaling-policy](#) Befehl mit der angegebenen Konfigurationsdatei aus, wie im folgenden Beispiel gezeigt.

```
aws autoscaling put-scaling-policy --policy-name alb1000-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er den Amazon-Ressourcennamen (ARN) der Richtlinie zurück.

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-
id:scalingPolicy:19556d63-7914-4997-8c81-d27ca5241386:autoScalingGroupName/my-
asg:policyName/alb1000-predictive-scaling-policy",
  "Alarms": []
}
```

Beispiel 3: Eine Richtlinie für die prädiktive Skalierung, die höher als die maximale Kapazität skaliert werden kann

Das folgende Beispiel zeigt, wie eine Richtlinie erstellt wird, die höher als die maximale Größenbeschränkung der Gruppe skaliert werden kann, wenn Sie eine höhere Last als die normale Last benötigen. Standardmäßig skaliert Amazon EC2 Auto Scaling Ihre EC2-Kapazität nicht höher als Ihre definierte maximale Kapazität. Es kann jedoch hilfreich sein, eine höhere Skalierung mit etwas mehr Kapazität zu ermöglichen, um Leistungs- oder Verfügbarkeitsprobleme zu vermeiden.

Um Amazon EC2 Auto Scaling Platz für die Bereitstellung zusätzlicher Kapazität zu bieten, wenn die Kapazität laut Prognose die maximale Größe Ihrer Gruppe erreichen oder ihr sehr nahe kommen wird, geben Sie die Eigenschaften `MaxCapacityBreachBehavior` und `MaxCapacityBuffer` wie im folgenden Beispiel gezeigt an. Sie müssen `MaxCapacityBreachBehavior` mit einem positiven Wert für `IncreaseMaxCapacity` angeben. Die maximale Anzahl von Instances, die Ihre Gruppe haben kann, hängt vom Wert von `MaxCapacityBuffer` ab.

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 70,
      "PredefinedMetricPairSpecification": {
        "PredefinedMetricType": "ASGCPUUtilization"
      }
    }
  ],
  "MaxCapacityBreachBehavior": "IncreaseMaxCapacity",
```

```
"MaxCapacityBuffer": 10  
}
```

In diesem Beispiel ist die Richtlinie so konfiguriert, dass sie einen 10-Prozent-Puffer ("MaxCapacityBuffer": 10) verwendet. Wenn die prognostizierte Kapazität also 50 und die maximale Kapazität 40 ist, ist die effektive maximale Kapazität 55. Eine Richtlinie, die eine Kapazität skalieren kann, die höher als die maximale Kapazität ist, um der prognostizierten Kapazität zu entsprechen, diese aber nicht zu überschreiten, hätte einen Puffer von 0 ("MaxCapacityBuffer": 0).

Um diese Richtlinie zu erstellen, führen Sie den [put-scaling-policy](#) Befehl mit der angegebenen Konfigurationsdatei aus, wie im folgenden Beispiel gezeigt.

```
aws autoscaling put-scaling-policy --policy-name cpu70-predictive-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \  
  --predictive-scaling-configuration file://config.json
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er den Amazon-Ressourcennamen (ARN) der Richtlinie zurück.

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:d02ef525-8651-4314-  
bf14-888331ebd04f:autoScalingGroupName/my-asg:policyName/cpu70-predictive-scaling-  
policy",  
  "Alarms": []  
}
```

Auswertung Ihrer Richtlinien für prädiktive Skalierung

Bevor Sie eine prädiktive Skalierungsrichtlinie zur Skalierung Ihrer Auto-Scaling-Gruppe verwenden, überprüfen Sie Ihre Empfehlungen und andere Daten zu Ihrer Richtlinie in der Konsole von Amazon EC2 Auto Scaling. Dies ist wichtig, denn eine Richtlinie für prädiktive Skalierung soll Ihre tatsächliche Kapazität erst dann skalieren, wenn Sie wissen, dass die Prognosen korrekt sind.

Wenn die Auto-Scaling-Gruppe neu ist, muss es mindestens 24 Stunden an Daten liefern, bevor Amazon EC2 Auto Scaling eine Prognose erstellen kann.

Wenn Amazon EC2 Auto Scaling eine Prognose erstellt, verwendet es Verlaufsdaten. Wenn Ihre Auto-Scaling-Gruppe nicht über ausreichend aktuelle Verlaufsdaten verfügt, füllt Amazon EC2

Auto Scaling die Prognose möglicherweise vorübergehend mit Aggregaten auf, die aus den aktuell verfügbaren Verlaufsaggregaten erstellt wurden. Prognosen werden bis zu zwei Wochen vor dem Erstellungsdatum einer Richtlinie aufgefüllt.

Inhalt

- [Anzeigen Ihrer Richtlinien für prädiktive Skalierung](#)
- [Anzeigen von Diagrammen zur Überwachung der prädiktiven Skalierung](#)
- [Überwachen Sie prädiktive Skalierungsmetriken mit CloudWatch](#)

Anzeigen Ihrer Richtlinien für prädiktive Skalierung

Für eine effektive Analyse sollte Amazon EC2 Auto Scaling über mindestens zwei Richtlinien für prädiktive Skalierung zum Vergleich verfügen. (Sie können die Ergebnisse jedoch weiterhin für eine einzelne Richtlinie überprüfen.) Wenn Sie mehrere Richtlinien erstellen, können Sie eine Richtlinie, die eine Metrik verwendet, gegen eine Richtlinie auswerten, die eine andere Metrik verwendet. Sie können auch die Auswirkungen verschiedener Zielwert- und Metrikkombinationen bewerten. Nachdem die Richtlinien für prädiktive Skalierung erstellt wurden, beginnt Amazon EC2 Auto Scaling sofort mit der Auswertung, welche Richtlinie für die Skalierung Ihrer Gruppe besser geeignet wäre.

So zeigen Sie Ihre Empfehlungen in der Konsole von Amazon EC2 Auto Scaling an

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Auf der Registerkarte Automatische Skalierung finden Sie unter Richtlinien für prädiktive Skalierung Details zu einer Richtlinie sowie unsere Empfehlung. In der Empfehlung erfahren Sie, ob es besser wäre, die Richtlinie für prädiktive Skalierung zu verwenden, oder nicht.

Wenn Sie sich nicht sicher sind, ob eine prädiktive Skalierungsrichtlinie für Ihre Gruppe geeignet ist, prüfen Sie die Spalten Auswirkungen auf die Verfügbarkeit und Auswirkungen auf die Kosten, um die richtige Richtlinie auszuwählen. Die Informationen in den einzelnen Spalten geben Aufschluss über die Auswirkungen der jeweiligen Richtlinie.

- **Auswirkungen auf die Verfügbarkeit:** Beschreibt, ob die Richtlinie negative Auswirkungen auf die Verfügbarkeit vermeiden würde, indem genügend Instances zur Bewältigung des

Workloads bereitgestellt werden, und zieht einen Vergleich für den Fall, dass die Richtlinie nicht verwendet wird.

- Auswirkungen auf die Kosten: Beschreibt, ob die Richtlinie negative Auswirkungen auf Ihre Kosten vermeiden würde, indem Instances nicht übermäßig bereitgestellt werden, und zieht einen Vergleich für den Fall, dass die Richtlinie nicht verwendet wird. Eine zu hohe Bereitstellung führt dazu, dass Ihre Instances nicht ausgelastet sind oder sich im Leerlauf befinden, was die Kosten nur noch weiter erhöht.

Wenn Sie über mehrere Richtlinien verfügen, wird neben dem Namen der Richtlinie, die die meisten Verfügbarkeitsvorteile zu geringeren Kosten bietet, ein Tag für die Beste Prognose angezeigt. Die Auswirkungen auf die Verfügbarkeit werden stärker gewichtet.

4. (Optional) Um den gewünschten Zeitraum für die Empfehlungsergebnisse auszuwählen, wählen Sie den gewünschten Wert aus der Dropdown-Liste Auswertungszeitraum: 2 Tage, 1 Woche, 2 Wochen, 4 Wochen, 6 Wochen oder 8 Wochen. Standardmäßig wird der Auswertungszeitraum auf die letzten zwei Wochen festgelegt. Ein längerer Auswertungszeitraum liefert mehr Datenpunkte für die Empfehlungsergebnisse. Das Hinzufügen weiterer Datenpunkte verbessert die Ergebnisse jedoch möglicherweise nicht, wenn sich Ihre Lastmuster geändert haben, z. B. nach einer Phase außergewöhnlich hoher Nachfrage. In diesem Fall können Sie eine gezieltere Empfehlung erhalten, indem Sie sich aktuellere Daten ansehen.

Note

Empfehlungen werden nur für Richtlinien generiert, die sich im Modus Nur Prognose befinden. Das Empfehlungs-Feature liefert bessere Ergebnisse, wenn sich eine Richtlinie während des gesamten Bewertungszeitraums im Modus Nur Prognose befindet. Wenn Sie eine Richtlinie im Prognose- und Skalierungsmodus starten und diese später in den Modus Nur Prognose wechselt, sind die Ergebnisse für diese Richtlinie wahrscheinlich verzerrt. Dies liegt daran, dass die Richtlinie bereits zur tatsächlichen Kapazität beigetragen hat.

Anzeigen von Diagrammen zur Überwachung der prädiktiven Skalierung

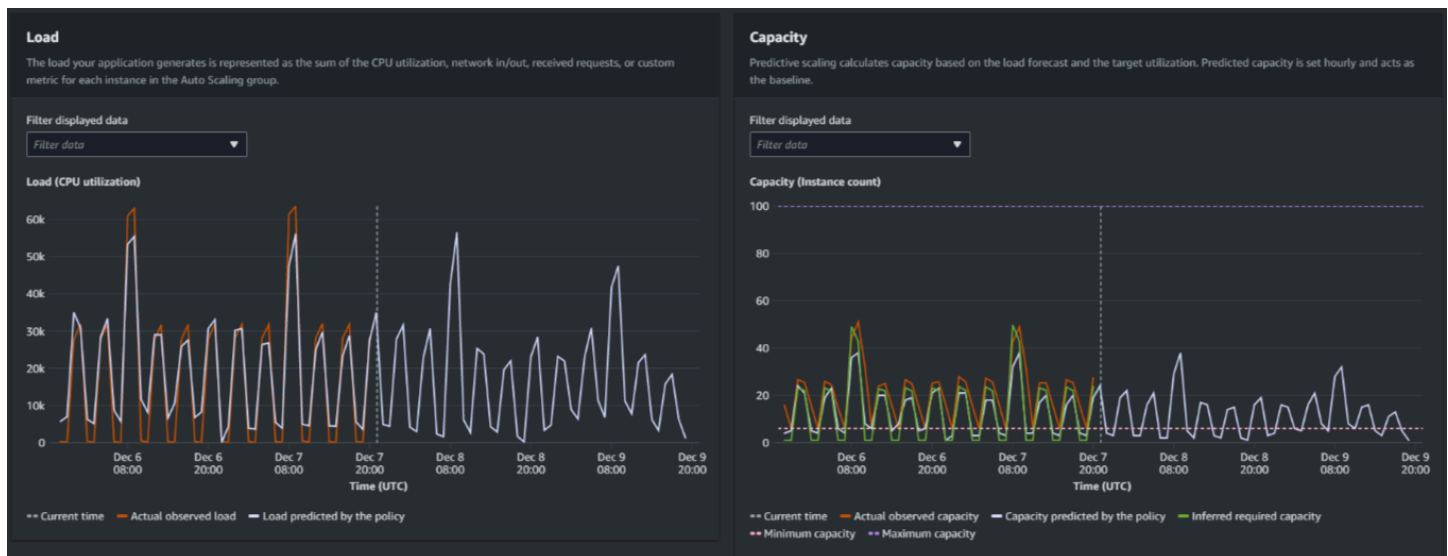
In der Konsole von Amazon EC2 Auto Scaling können Sie die Prognose der vergangenen Tage, Wochen oder Monate überprüfen, um zu visualisieren, wie gut die Richtlinie im Laufe der Zeit funktioniert. Sie können diese Informationen auch zur Auswertung der Genauigkeit von Vorhersagen

verwenden, wenn Sie entscheiden, ob Sie Ihre tatsächliche Kapazität durch eine Richtlinie skalieren lassen möchten.

So überprüfen Sie Überwachungsdiagramme für die prädiktive Skalierung in der Konsole von Amazon EC2 Auto Scaling

1. Wählen Sie eine Richtlinie aus der Liste Richtlinien für prädiktive Skalierung aus.
2. Im Abschnitt Überwachung können Sie die vergangenen und zukünftigen Prognosen Ihrer Richtlinie für Last und Kapazität im Vergleich zu tatsächlichen Werten anzeigen. Das Diagramm für Last zeigt Auslastungsprognosen und tatsächliche Werte für die ausgewählte Auslastungsmetrik. Das Diagramm zur Kapazität zeigt die Anzahl der von der Richtlinie vorhergesagten Instances. Es enthält auch die tatsächliche Anzahl der gestarteten Instances. Die vertikale Linie trennt Verlaufswerte von zukünftigen Prognosen. Diese Diagramme stehen kurz nach der Erstellung der Richtlinie zur Verfügung.
3. (Optional) Um die Menge der im Diagramm angezeigten Verlaufsdaten zu ändern, wählen Sie Ihren bevorzugten Wert aus der Dropdown-Liste Auswertungszeitraum oben auf der Seite aus. Der Auswertungszeitraum verändert die Daten auf dieser Seite in keiner Weise. Er ändert nur die Menge der angezeigten Verlaufsdaten.

Das folgende Image zeigt die Diagramme für Last und Kapazität, wenn Prognosen mehrfach angewendet wurden. Die prädiktive Skalierung prognostiziert die Last basierend auf Ihren historischen Lastdaten. Die von Ihrer Anwendung erzeugte Last wird als Summe der CPU-Auslastung, des Netzwerkeingangs/-ausgangs, der empfangenen Anfragen oder der benutzerdefinierten Metrik für jede Instance in der Auto-Scaling-Gruppe dargestellt. Die prädiktive Skalierung berechnet den zukünftigen Kapazitätsbedarf basierend auf der Lastprognose und der Zielauslastung, die Sie für die Skalierungsmetrik erreichen möchten.



Vergleichen von Daten im Diagramm für Last

Jede horizontale Linie stellt einen anderen Satz von Datenpunkten dar, die in einstündigen Intervallen gemeldet werden:

1. Die tatsächlich beobachtete Last verwendet die SUM-Statistik für die von Ihnen gewählte Lastmetrik, um die vergangene stündliche Gesamtlast anzuzeigen.
2. Von der Richtlinie prognostizierte Last zeigt die stündliche Lastprognose. Diese Prognose basiert auf den tatsächlichen Lastbeobachtungen der letzten zwei Wochen.

Vergleichen von Daten im Diagramm zur Kapazität

Jede horizontale Linie stellt einen anderen Satz von Datenpunkten dar, die in einstündigen Intervallen gemeldet werden:

1. Die tatsächliche beobachtete Kapazität zeigt die tatsächliche Kapazität Ihrer Auto Scaling Scaling-Gruppe in der Vergangenheit an, wenn die [GroupTotalInstances](#) Metrik aktiviert ist. Diese Kapazität hängt von Ihren anderen Skalierungsrichtlinien und der Mindestgruppengröße während des ausgewählten Zeitraums ab.
2. Von der Richtlinie prognostizierte Kapazität zeigt die Basiskapazität an, die Sie zu Beginn jeder Stunde erwarten können, wenn sich die Richtlinie im Modus Prognose und Skalierung befindet.
3. Abgeleitete erforderliche Kapazität zeigt die ideale Kapazität, um die Skalierungsmetrik auf dem von Ihnen gewählten Zielwert zu halten.
4. Mindestkapazität zeigt die Mindestkapazität der Auto-Scaling-Gruppe an.

5. Maximale Kapazität zeigt die maximale Kapazität der Auto-Scaling-Gruppe.

Um die abgeleitete erforderliche Kapazität zu berechnen, gehen wir zunächst davon aus, dass jede Instance bei einem bestimmten Zielwert gleichmäßig ausgelastet ist. In der Praxis werden Instances nicht gleichmäßig ausgelastet. Wenn wir jedoch davon ausgehen, dass die Auslastung gleichmäßig auf die Instances verteilt ist, können wir eine wahrscheinliche Schätzung der benötigten Kapazität vornehmen. Der Kapazitätsbedarf wird dann umgekehrt proportional zu der Skalierungsmetrik berechnet, die Sie für Ihre Richtlinie für prädiktive Skalierung verwendet haben. Mit anderen Worten heißt das: Wenn die Kapazität zunimmt, nimmt die Skalierungsmetrik im gleichen Maß ab. Wenn sich beispielsweise die Kapazität verdoppelt, muss die Skalierungsmetrik um die Hälfte verringert werden.

Die Formel für die abgeleitete erforderliche Kapazität lautet wie folgt:

$$\text{sum of } (\text{actualCapacityUnits} * \text{scalingMetricValue}) / (\text{targetUtilization})$$

Als Beispiel nehmen wir den `actualCapacityUnits` (10) und den `scalingMetricValue` (30) für eine bestimmte Stunde her. Wir nehmen dann die `targetUtilization`, die Sie in Ihrer Richtlinie für prädiktive Skalierung (60) angegeben haben, und berechnen die abgeleitete erforderliche Kapazität für dieselbe Stunde. Dies gibt den Wert 5 zurück. Das bedeutet, dass fünf die abgeleitete Kapazität ist, die erforderlich ist, um die Kapazität im direkt umgekehrten Verhältnis zum Zielwert der Skalierungsmetrik zu erhalten.

Note

Es stehen Ihnen verschiedene Möglichkeiten zur Verfügung, mit denen Sie die Kosteneinsparungen und die Verfügbarkeit Ihrer Anwendung verbessern können.

- Sie verwenden die prädiktive Skalierung für die Basiskapazität und die dynamische Skalierung für den Umgang mit zusätzlicher Kapazität. Die dynamische Skalierung funktioniert unabhängig von der prädiktiven Skalierung, indem sie basierend auf der aktuellen Auslastung ab- und aufskaliert. Zunächst berechnet Amazon EC2 Auto Scaling die empfohlene Anzahl von Instances für jede dynamische Skalierungsrichtlinie. Anschließend skaliert die Lösung basierend auf der Richtlinie, die die größte Anzahl von Instances bereitstellt.
- Damit bei sinkender Last eine Abskalierung erfolgen kann, sollte Ihre Auto-Scaling-Gruppe immer über mindestens eine dynamische Skalierungsrichtlinie verfügen, bei der das Abskalieren aktiviert ist.

- Sie können die Skalierungsleistung verbessern, indem Sie sicherstellen, dass Ihre Mindest- und Höchstkapazität nicht zu restriktiv ist. Eine Richtlinie mit einer empfohlenen Anzahl von Instances, die nicht innerhalb des Mindest- und Höchstkapazitätsbereichs liegt, wird an der Ab- und Aufskalierung gehindert.

Überwachen Sie prädiktive Skalierungsmetriken mit CloudWatch

Je nach Ihren Anforderungen ziehen Sie es möglicherweise vor, auf Überwachungsdaten für die prädiktive Skalierung von Amazon CloudWatch statt über die Amazon EC2 Auto Scaling Scaling-Konsole zuzugreifen. Nach Erstellung einer prädiktiven Skalierungsrichtlinie werden Daten gesammelt, um Ihre zukünftige Last und Kapazität zu prognostizieren. Nachdem diese Daten erfasst wurden, werden sie automatisch in regelmäßigen CloudWatch Abständen gespeichert. Anschließend können Sie visualisieren CloudWatch, wie gut die Richtlinie im Laufe der Zeit funktioniert. Sie können auch CloudWatch Alarme erstellen, um Sie zu benachrichtigen, wenn sich Leistungsindikatoren über die von Ihnen definierten Grenzwerte hinaus ändern CloudWatch.

Topics

- [Visualisieren historischer Prognosedaten](#)
- [Erstellen von Genauigkeitsmetriken mithilfe von Metrikberechnungen](#)

Visualisieren historischer Prognosedaten

Die Last- und Kapazitätsprognosedaten für eine Richtlinie zur vorausschauenden Skalierung finden Sie unter CloudWatch. Dies kann nützlich sein, wenn Sie Prognosen im Vergleich zu anderen CloudWatch Kennzahlen in einem einzigen Diagramm visualisieren möchten. Es kann auch hilfreich sein, wenn Sie einen größeren Zeitraum anzeigen, um Trends im Zeitverlauf zu erkennen. Ihnen stehen historische Metriken von bis zu 15 Monaten zur Verfügung, um die Leistung Ihrer Richtlinie besser analysieren zu können.

Weitere Informationen finden Sie unter [Metriken und Dimensionen für die prädiktive Skalierung](#).

Um historische Prognosedaten mit der CloudWatch Konsole anzuzeigen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Metrics (Metriken) und dann All metrics (Alle Metriken) aus.
3. Wählen Sie Metrik-namespace Auto Scaling aus.

4. Wählen Sie eine der folgenden Optionen aus, um entweder die Lastprognose- oder die Kapazitätsprognosemetriken anzuzeigen:
 - Prädiktive Skalierung: Lastprognosen
 - Prädiktive Skalierung: Kapazitätsprognosen
5. Geben Sie im Suchfeld den Namen der prädiktiven Skalierungsrichtlinie oder den Namen der Auto-Scaling-Gruppe ein, und drücken Sie dann die Eingabetaste, um die Ergebnisse zu filtern.
6. Um eine Metrik grafisch darzustellen, müssen Sie das Kontrollkästchen neben der Metrik aktivieren. Wenn Sie den Namen des Diagramms ändern möchten, wählen Sie das Bleistiftsymbol. Wenn Sie den Zeitraum ändern möchten, müssen Sie einen der vordefinierten Werte oder custom (benutzerdefiniert) auswählen. Weitere Informationen finden Sie unter [Grafische Darstellung einer Metrik](#) im CloudWatch Amazon-Benutzerhandbuch.
7. Wenn Sie die Statistik ändern möchten, wählen Sie die Registerkarte Graphed metrics aus. Wählen Sie die Spaltenüberschrift oder einen einzelnen Wert und anschließend eine andere Statistik aus. Sie können zwar für jede Metrik eine beliebige Statistik wählen, aber nicht alle Statistiken sind für PredictiveScalingLoadForecastMetriken PredictiveScalingCapacityForecastnützlich. So sind zum Beispiel die Statistiken Durchschnitt, Minimum und Maximum hilfreich, die Statistik Summe jedoch nicht.
8. Wenn Sie dem Diagramm eine weitere Metrik hinzufügen möchten, wählen Sie unter Browse (Durchsuchen) die Option All (Alle) aus, suchen Sie nach der spezifischen Metrik, und aktivieren Sie dann das zugehörige Kontrollkästchen. Sie können bis zu 10 Metriken hinzufügen.

Wenn Sie dem Diagramm beispielsweise die tatsächlichen Werte für die CPU-Auslastung hinzufügen möchten, wählen Sie den Namespace EC2 und anschließend By Auto Scaling Group (Nach Auto-Scaling-Gruppe) aus. Aktivieren Sie dann das Kontrollkästchen für die CPUUtilizationMetrik und die spezifische Auto Scaling Scaling-Gruppe.

9. (Optional) Um das Diagramm zu einem CloudWatch Dashboard hinzuzufügen, wählen Sie Aktionen, Zum Dashboard hinzufügen aus.

Erstellen von Genauigkeitsmetriken mithilfe von Metrikberechnungen

Mit metrischer Mathematik können Sie mehrere CloudWatch Metriken abfragen und mathematische Ausdrücke verwenden, um neue Zeitreihen auf der Grundlage dieser Metriken zu erstellen. Sie können die resultierenden Zeitreihen auf der CloudWatch Konsole visualisieren und sie zu Dashboards hinzufügen. Weitere Informationen zur metrischen Mathematik finden Sie unter [Verwenden von metrischer Mathematik](#) im CloudWatch Amazon-Benutzerhandbuch.

Mithilfe von Metrikberechnungen können Sie die Daten, die Amazon EC2 Auto Scaling für die prädiktive Skalierung generiert, auf unterschiedliche Weise grafisch darstellen. So können Sie die Leistung von Richtlinien im Zeitverlauf überwachen und erkennen, ob Ihre Kombination von Metriken möglicherweise verbessert werden kann.

Sie können beispielsweise einen Metrikberechnungsausdruck verwenden, um den [Mean Absolute Percentage Error](#) (MAPE) zu überwachen. Die MAPE-Metrik hilft bei der Überwachung der Differenz zwischen den prognostizierten Werten und den tatsächlichen Werten eines bestimmten Prognosefensters. Änderungen des MAPE-Werts können Aufschluss darüber geben, ob sich die Leistung der Richtlinie im Laufe der Zeit verschlechtert, wenn sich Ihre Anwendung verändert. Eine Erhöhung des MAPE-Werts bedeutet eine größere Diskrepanz zwischen prognostizierten und tatsächlichen Werten.

Beispiel: Metrikberechnungsausdruck

Für die ersten Schritte mit dieser Art von Diagramm können Sie beispielsweise den Metrikberechnungsausdruck aus dem folgenden Beispiel erstellen.

```
{
  "MetricDataQueries": [
    {
      "Expression": "TIME_SERIES(AVG(ABS(m1-m2)/m1))",
      "Id": "e1",
      "Period": 3600,
      "Label": "MeanAbsolutePercentageError",
      "ReturnData": true
    },
    {
      "Id": "m1",
      "Label": "ActualLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/EC2",
          "MetricName": "CPUUtilization",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            }
          ]
        }
      }
    },
    {
      "Period": 3600,
```

```

    "Stat": "Sum"
  },
  "ReturnData": false
},
{
  "Id": "m2",
  "Label": "ForecastedLoadValues",
  "MetricStat": {
    "Metric": {
      "Namespace": "AWS/AutoScaling",
      "MetricName": "PredictiveScalingLoadForecast",
      "Dimensions": [
        {
          "Name": "AutoScalingGroupName",
          "Value": "my-asg"
        },
        {
          "Name": "PolicyName",
          "Value": "my-predictive-scaling-policy"
        },
        {
          "Name": "PairIndex",
          "Value": "0"
        }
      ]
    },
    "Period": 3600,
    "Stat": "Average"
  },
  "ReturnData": false
}
]
}

```

Anstelle einer einzelnen Metrik gibt es für `MetricDataQueries` ein Array von Abfragestrukturen für Metrikdaten. Jedes Element in `MetricDataQueries` ruft eine Metrik ab oder wendet einen mathematischen Ausdruck an. Das erste Element (`e1`) ist der mathematische Ausdruck. Der angegebene Ausdruck legt den Parameter `ReturnData` auf `true` fest, was letztendlich eine einzelne Zeitreihe generiert. Für alle anderen Metriken hat `ReturnData` den Wert `false`.

In diesem Beispiel verwendet der angegebene Ausdruck die tatsächlichen und prognostizierten Werte als Eingabe und gibt die neue Metrik (MAPE) zurück. `m1` ist die CloudWatch Metrik, die

die tatsächlichen Lastwerte enthält (vorausgesetzt, die CPU-Auslastung ist die Lastmetrik, die ursprünglich für die genannte `my-predictive-scaling-policy` Richtlinie angegeben wurde). `m2` ist die CloudWatch Metrik, die die prognostizierten Lastwerte enthält. Die mathematische Syntax für die MAPE-Metrik lautet wie folgt:

Durchschnitt von $(\text{abs}((\text{tatsächlicher Wert} - \text{prognostizierter Wert})/(\text{tatsächlichen Wert})))$

Visualisieren Ihrer Genauigkeitsmetriken und Festlegen von Alarmen

Um die Genauigkeitsmetrikdaten zu visualisieren, wählen Sie in der CloudWatch Konsole die Registerkarte Metriken aus. Von dort aus können Sie die Daten grafisch darstellen. Weitere Informationen finden Sie unter [Hinzufügen eines mathematischen Ausdrucks zu einem CloudWatch Diagramm](#) im CloudWatch Amazon-Benutzerhandbuch.

Im Abschnitt Metrics (Metriken) können Sie auch einen Alarm für eine von Ihnen überwachte Metrik festlegen. Wählen Sie auf der Registerkarte Graphed metrics (Grafisch dargestellte Metriken) unter der Spalte Actions (Aktionen) das Symbol Create alarm (Alarm erstellen) aus. Das Symbol Create alarm (Alarm erstellen) wird als kleine Glocke dargestellt. Weitere Informationen und Benachrichtigungsoptionen finden Sie unter [Erstellen eines CloudWatch Alarms auf der Grundlage eines metrischen mathematischen Ausdrucks](#) und [Benachrichtigung von Benutzern über Alarmänderungen](#) im CloudWatch Amazon-Benutzerhandbuch.

Alternativ können Sie [GetMetricData](#) verwenden, [PutMetricAlarm](#) um Berechnungen mithilfe metrischer Mathematik durchzuführen und Alarme auf der Grundlage der Ausgabe zu erstellen.

Überschreiben von Prognosewerten mithilfe geplanter Aktionen

Manchmal haben Sie möglicherweise zusätzliche Informationen zu Ihren zukünftigen Anwendungsanforderungen, die bei der Prognoseberechnung nicht berücksichtigt werden können. Prognoseberechnungen können beispielsweise die Kapazität unterschätzen, die für eine bevorstehende Marketingveranstaltung benötigt wird. Sie können geplante Aktionen verwenden, um die Prognose in zukünftigen Zeiträumen vorübergehend zu überschreiben. Die geplanten Aktionen können auf einer wiederkehrenden Basis oder zu einem bestimmten Zeitpunkt ausgeführt werden, wenn einmalige Nachfrageschwankungen auftreten.

Sie können beispielsweise eine geplante Aktion mit einer höheren Mindestkapazität als die prognostizierte Aktion erstellen. Zur Laufzeit aktualisiert Amazon EC2 Auto Scaling die minimale Kapazität Ihrer Auto-Scaling-Gruppe. Da die prädiktive Skalierung für die Kapazität optimiert wird, wird eine geplante Aktion mit einer minimalen Kapazität, die höher als die Prognosewerte

ist, berücksichtigt. Dadurch wird verhindert, dass die Kapazität geringer ist als erwartet. Um das Überschreiben der Prognose zu beenden, setzen Sie über eine zweite geplante Aktion die minimale Kapazität auf ihre ursprüngliche Einstellung zurück.

Im folgenden Verfahren werden die Schritte zum Überschreiben der Prognose in zukünftigen Zeiträumen erläutert.

Themen

- [Schritt 1: \(Optional\) Analysieren von Zeitreihendaten](#)
- [Schritt 2: Erstellen von zwei geplanten Aktionen](#)

Important

In diesem Thema wird davon ausgegangen, dass Sie versuchen, die Prognose zu überschreiben, um auf eine höhere Kapazität als die prognostizierte zu skalieren. Wenn Sie die Kapazität vorübergehend verringern müssen, ohne dass dies durch eine Richtlinie zur vorausschauenden Skalierung beeinträchtigt wird, verwenden Sie stattdessen den Modus „Nur Prognosen“. Im Modus „Nur Prognose“ generiert die vorausschauende Skalierung zwar weiterhin Prognosen, erhöht aber nicht automatisch die Kapazität. Anschließend können Sie die Ressourcennutzung überwachen und die Größe Ihrer Gruppe nach Bedarf manuell verringern. Weitere Informationen zur manuellen Skalierung finden Sie unter [Manuelle Skalierung für Amazon EC2 Auto Scaling](#).

Schritt 1: (Optional) Analysieren von Zeitreihendaten

Beginnen Sie mit der Analyse der Prognose-Zeitreehendaten. Dies ist ein optionaler Schritt, aber es ist hilfreich, wenn Sie die Details der Prognose verstehen möchten.

1. Rufen Sie die Prognose ab

Nachdem die Prognose erstellt wurde, können Sie einen bestimmten Zeitraum in der Prognose abfragen. Ziel der Abfrage ist es, einen vollständigen Überblick über die Zeitreehendaten für einen bestimmten Zeitraum zu erhalten.

Ihre Abfrage kann Prognosedaten bis zwei Tage in die Zukunft enthalten. Wenn Sie die prädiktive Skalierung eine Weile verwenden, können Sie auch auf Ihre früheren Prognosedaten zugreifen. Der maximale Zeitraum zwischen der Start- und Endzeit beträgt jedoch 30 Tage.

Um die Prognose mithilfe des [get-predictive-scaling-forecast](#) AWS CLI Befehls abzurufen, geben Sie im Befehl die folgenden Parameter an:

- Geben Sie den Namen der Auto-Scaling-Gruppe im Feld `--auto-scaling-group-name`-Parameter an.
- Geben Sie den Namen der Richtlinie im `--policy-name`-Parameter an.
- Geben Sie die Startzeit im `--start-time`-Parameter an, um nur prognostizierte Daten für den angegebenen Zeitpunkt oder danach zurückzugeben.
- Geben Sie die Endzeit im `--end-time`-Parameter an, um nur prognostizierte Daten für den angegebenen Zeitpunkt oder davor zurückzugeben.

```
aws autoscaling get-predictive-scaling-forecast --auto-scaling-group-name my-asg \  
--policy-name cpu40-predictive-scaling-policy \  
--start-time "2021-05-19T17:00:00Z" \  
--end-time "2021-05-19T23:00:00Z"
```

Bei erfolgreicher Ausführung gibt der Befehl Daten zurück, die in etwa wie folgt aussehen:

```
{  
  "LoadForecast": [  
    {  
      "Timestamps": [  
        "2021-05-19T17:00:00+00:00",  
        "2021-05-19T18:00:00+00:00",  
        "2021-05-19T19:00:00+00:00",  
        "2021-05-19T20:00:00+00:00",  
        "2021-05-19T21:00:00+00:00",  
        "2021-05-19T22:00:00+00:00",  
        "2021-05-19T23:00:00+00:00"  
      ],  
      "Values": [  
        153.0655799339254,  
        128.8288551285919,  
        107.1179447150675,  
        197.3601844551528,  
        626.4039934516954,  
        596.9441277518481,  
        677.9675713779869  
      ]  
    }  
  ],  
}
```

```

        "MetricSpecification": {
            "TargetValue": 40.0,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ASGCPUUtilization"
            }
        }
    ],
    "CapacityForecast": {
        "Timestamps": [
            "2021-05-19T17:00:00+00:00",
            "2021-05-19T18:00:00+00:00",
            "2021-05-19T19:00:00+00:00",
            "2021-05-19T20:00:00+00:00",
            "2021-05-19T21:00:00+00:00",
            "2021-05-19T22:00:00+00:00",
            "2021-05-19T23:00:00+00:00"
        ],
        "Values": [
            2.0,
            2.0,
            2.0,
            2.0,
            4.0,
            4.0,
            4.0
        ]
    },
    "UpdateTime": "2021-05-19T01:52:50.118000+00:00"
}

```

Die Antwort enthält zwei Prognosen: `LoadForecast` und `CapacityForecast`. `LoadForecast` zeigt die stündliche Lastprognose an. `CapacityForecast` zeigt Prognosewerte für die Kapazität an, die stündlich benötigt wird, um die prognostizierte Last zu verarbeiten, während ein `TargetValue` von 40,0 (40 % durchschnittliche CPU-Auslastung) aufrechterhalten bleibt.

2. Identifizieren des Zielzeitraums

Ermitteln Sie die Stunde oder die Stunden, zu der/zu denen die einmalige Nachfrageschwankung stattfinden soll. Denken Sie daran, dass die in der Prognose angezeigten Datumsangaben und Uhrzeiten in UTC angegeben sind.

Schritt 2: Erstellen von zwei geplanten Aktionen

Erstellen Sie als Nächstes zwei geplante Aktionen für einen bestimmten Zeitraum, in dem Ihre Anwendung eine höhere Last aufweist als die prognostizierte Last. Wenn Sie beispielsweise während eines Marketing-Ereignisses für einen begrenzten Zeitraum ein erhöhtes Daten-Volumen erwarten, können Sie eine einmalige Aktion planen, um die Mindestkapazität bei deren Beginn zu aktualisieren. Planen Sie dann eine weitere Aktion, um die Mindestkapazität auf die ursprüngliche Einstellung zurückzusetzen, wenn das Ereignis endet.

Erstellen von zwei geplanten Aktionen für einmalige Ereignisse (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Automatic scaling (Automatische Skalierung) unter Scheduled actions (Geplante Aktionen) die Option Geplante Aktion erstellen (Create scheduled action) aus.
4. Geben Sie die folgenden Einstellungen für die geplante Aktion ein:
 - a. Geben Sie einen Namen für die geplante Aktion ein.
 - b. Für Min geben Sie die neue Mindestkapazität für Ihre Auto-Scaling-Gruppe ein. Der Min-Wert darf maximal so groß sein wie die Höchstgröße der Gruppe. Wenn Ihr Wert für Min größer ist als die Höchstgröße der Gruppe, müssen Sie Max aktualisieren.
 - c. Wählen Sie für Recurrence (Wiederholung) Once (Einmal) aus.
 - d. Wählen Sie für Zeitzone eine Zeitzone aus. Wenn keine Zeitzone gewählt ist, wird standardmäßig ETC/UTC verwendet.
 - e. Definieren Sie eine Spezifische Startzeit.
5. Wählen Sie Erstellen aus.

Die Konsole zeigt die geplanten Aktionen der Auto-Scaling-Gruppe an.

6. Konfigurieren Sie eine zweite geplante Aktion, damit die Mindestkapazität am Ende des Ereignisses wieder auf die ursprüngliche Einstellung zurückkehrt. Die prädiktive Skalierung kann die Kapazität nur skalieren, wenn der Wert, den Sie für Min angeben, niedriger ist als die Prognosewerte.

Erstellen von zwei geplanten Aktionen für einmalige Ereignisse (AWS CLI)

Verwenden Sie den AWS CLI Befehl [put-scheduled-update-group-action](#), um die geplanten Aktionen zu erstellen.

Lassen Sie uns als Beispiel einen Zeitplan definieren, der am 19. Mai um 17:00 Uhr acht Stunden lang eine Mindestkapazität von drei Instances beibehält. Die folgenden Befehle veranschaulichen die Implementierung dieses Szenarios.

Der erste Befehl [put-scheduled-update-group-action](#) weist Amazon EC2 Auto Scaling an, die Mindestkapazität der angegebenen Auto Scaling-Gruppe am 19. Mai 2021 um 17:00 Uhr UTC zu aktualisieren.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-start \  
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-  
capacity 3
```

Der zweite Befehl weist Amazon EC2 Auto Scaling an, die Mindestkapazität der Gruppe am 20. Mai 2021 um 1:00 Uhr UTC auf eins zu setzen.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end \  
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-  
capacity 1
```

Nachdem Sie der Auto-Scaling-Gruppe diese geplanten Aktionen hinzugefügt haben, führt Amazon EC2 Auto Scaling folgende Schritte aus:

- Um 17:00 Uhr UTC am 19. Mai 2021 wird die erste geplante Aktion ausgeführt. Wenn die Gruppe derzeit weniger als drei Instances hat, wird die Gruppe auf drei Instances skaliert. Während dieser Zeit und für die Dauer der nächsten acht Stunden kann Amazon EC2 Auto Scaling weiterhin skaliert werden, wenn die prognostizierte Kapazität höher als die tatsächliche Kapazität ist oder wenn eine Richtlinie für dynamische Skalierung in Kraft ist.
- Um 1:00 Uhr UTC am 20. Mai 2021 wird die zweite geplante Aktion ausgeführt. Dadurch wird die Mindestkapazität am Ende des Ereignisses auf die ursprüngliche Einstellung zurückgesetzt.

Skalierung basierend auf wiederkehrenden Zeitplänen

Um die Prognose jede Woche während des gleichen Zeitraums zu überschreiben, erstellen Sie zwei geplante Aktionen und stellen die Zeit- und Datumslogik mithilfe eines Cron-Ausdrucks bereit.

Der Cron-Ausdruck besteht aus fünf Feldern, getrennt durch Leerzeichen: [Minute] [Stunde] [Tag_des_Monats] [Monat_des_Jahres] [Wochentag]. Felder können alle zulässigen Werte enthalten, einschließlich Sonderzeichen.

Beispielsweise führt der folgende Cron-Ausdruck jeden Dienstag um 6:30 Uhr die Aktion aus. Das Sternchen wird als Platzhalter verwendet, um alle Werte für ein Feld abzugleichen.

```
30 6 * * 2
```

Weitere Informationen finden Sie auch unter

Weitere Informationen zum Erstellen, Auflisten, Bearbeiten und Löschen von geplanten Aktionen finden Sie unter [Geplante Skalierung für Amazon EC2 Auto Scaling](#).

Erweiterte prädiktive Skalierungsrichtlinie unter Verwendung benutzerdefinierter Metriken

In einer prädiktiven Skalierungsrichtlinie können Sie vordefinierte oder benutzerdefinierte Metriken verwenden. Benutzerdefinierte Metriken sind nützlich, wenn die vordefinierten Metriken (CPU, Netzwerk-I/O und Anzahl der Anfragen an den Application Load Balancer) Ihre Anwendungslast nicht ausreichend beschreiben.

Wenn Sie eine Richtlinie für vorausschauende Skalierung mit benutzerdefinierten Metriken erstellen, können Sie andere CloudWatch Messwerte angeben, die von bereitgestellt werden AWS, oder Sie können Metriken angeben, die Sie selbst definieren und veröffentlichen. Sie können auch metrische Mathematik verwenden, um bestehende Metriken zu aggregieren und in eine neue Zeitreihe umzuwandeln, die AWS nicht automatisch erfasst wird. Wenn Sie Werte in Ihren Daten kombinieren, indem Sie z.B. neue Summen oder Durchschnittswerte berechnen, nennt man das Aggregieren. Die resultierenden Daten werden als Aggregat bezeichnet.

Der folgende Abschnitt enthält bewährte Verfahren und Beispiele für die Erstellung der JSON-Struktur für die Richtlinie.

Themen

- [Best Practices](#)
- [Voraussetzungen](#)
- [Konstruieren von JSON für benutzerdefinierte Metriken](#)
- [Überlegungen zu benutzerdefinierten Metriken in einer Richtlinie zur prädiktiven Skalierung](#)
- [Einschränkungen](#)

Best Practices

Die folgenden bewährten Methoden können Ihnen helfen, benutzerdefinierte Metriken effektiver zu nutzen:

- Für die Spezifikation der Lastmetrik ist die nützlichste Metrik eine Metrik, die die Last einer Auto-Scaling-Gruppe als Ganzes darstellt, unabhängig von der Kapazität der Gruppe.
- Bei der Angabe der Skalierungsmetrik ist die sinnvollste Metrik für die Skalierung ein durchschnittlicher Durchsatz oder eine durchschnittliche Auslastung pro Instance.
- Die Skalierungsmetrik muss umgekehrt proportional zur Kapazität sein. Das heißt, wenn die Anzahl der Instances in der Auto-Scaling-Gruppe steigt, sollte die Skalierungsmetrik in etwa im gleichen Verhältnis sinken. Um sicherzustellen, dass sich die prädiktive Skalierung wie erwartet verhält, müssen die Lastmetrik und die Skalierungsmetrik auch stark miteinander korrelieren.
- Die Zielauslastung muss mit der Art der Skalierungsmetrik übereinstimmen. Bei einer Richtlinienkonfiguration, die die CPU-Auslastung verwendet, ist dies ein Zielprozentsatz. Bei einer Richtlinienkonfiguration, die den Durchsatz verwendet, wie z.B. die Anzahl der Anfragen oder Nachrichten, ist dies die angestrebte Anzahl von Anfragen oder Nachrichten pro Instance während eines einminütigen Intervalls.
- Wenn diese Empfehlungen nicht befolgt werden, werden die prognostizierten zukünftigen Werte der Zeitreihen wahrscheinlich falsch sein. Um zu überprüfen, ob die Daten korrekt sind, können Sie die prognostizierten Werte in der Amazon EC2 Auto Scaling-Konsole einsehen. Alternativ können Sie nach der Erstellung Ihrer Richtlinie für vorausschauende Skalierung die `CapacityForecast` Objekte `LoadForecast` und überprüfen, die durch einen [GetPredictiveScalingForecast](#) API-Aufruf zurückgegeben wurden.
- Wir empfehlen Ihnen dringend, die prädiktive Skalierung im Modus "Nur Prognose" zu konfigurieren, damit Sie die Prognose auswerten können, bevor die prädiktive Skalierung mit der aktiven Skalierung der Kapazität beginnt.

Voraussetzungen

Um benutzerdefinierte Metriken zu Ihrer prädiktiven Skalierungsrichtlinie hinzuzufügen, müssen Sie über entsprechende `cloudwatch:GetMetricData`-Berechtigungen verfügen.

Wenn Sie Ihre eigenen Metriken anstelle der bereitgestellten Metriken angeben möchten, müssen Sie Ihre Metriken zunächst auf CloudWatch veröffentlichen. AWS Weitere Informationen finden Sie unter [Veröffentlichen benutzerdefinierter Metriken](#) im CloudWatch Amazon-Benutzerhandbuch.

Sollten Sie Ihre eigenen Metriken veröffentlichen, achten Sie darauf, dass Sie die Datenpunkte mindestens alle fünf Minuten veröffentlichen. Amazon EC2 Auto Scaling ruft die Datenpunkte CloudWatch basierend auf der Länge des benötigten Zeitraums ab. Beispielsweise verwendet die Lastmetrikspezifikation stündliche Metriken, um die Auslastung Ihrer Anwendung zu messen. CloudWatch verwendet Ihre veröffentlichten Metrikdaten, um einen einzelnen Datenwert für einen beliebigen Zeitraum von einer Stunde bereitzustellen, indem alle Datenpunkte mit Zeitstempeln aggregiert werden, die in jeden Zeitraum von einer Stunde fallen.

Konstruieren von JSON für benutzerdefinierte Metriken

Der folgende Abschnitt enthält Beispiele für die Konfiguration der prädiktiven Skalierung für die Abfrage von Daten. CloudWatch Es gibt zwei verschiedene Methoden, um diese Option zu konfigurieren, und die von Ihnen gewählte Methode wirkt sich darauf aus, welches Format Sie verwenden, um den JSON für Ihre prädiktive Skalierungsrichtlinie zu erstellen. Wenn Sie metrische Berechnungen verwenden, variiert das Format von JSON je nach der durchgeführten metrischen Berechnung weiter.

1. Informationen zum Erstellen einer Richtlinie, mit der Daten direkt aus anderen CloudWatch Metriken abgerufen werden, die von bereitgestellt werden AWS oder für die Sie Daten veröffentlichen CloudWatch, finden [Beispiel einer prädiktiven Skalierungsrichtlinie mit benutzerdefinierten Last- und Skalierungsmetriken \(AWS CLI\)](#) Sie unter.
2. Informationen zum Erstellen einer Richtlinie, mit der mehrere CloudWatch Messwerte abgefragt und mithilfe mathematischer Ausdrücke neue Zeitreihen auf der Grundlage dieser Messwerte erstellt werden können, finden Sie unter [Metrikberechnungs-Ausdrücke verwenden](#).

Beispiel einer prädiktiven Skalierungsrichtlinie mit benutzerdefinierten Last- und Skalierungsmetriken (AWS CLI)

Um eine prädiktive Skalierungsrichtlinie mit benutzerdefinierten Last- und Skalierungsmetriken mit dem zu erstellen AWS CLI, speichern Sie die Argumente für `--predictive-scaling-configuration` in einer JSON-Datei mit dem Namen `config.json`.

Sie beginnen mit dem Hinzufügen benutzerdefinierter Metriken, indem Sie die ersetzbaren Werte im folgenden Beispiel durch die Werte Ihrer Metriken und Ihrer Zielauslastung ersetzen.

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Average"
            }
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyLoadMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
```

```
{
  {
    "Name": "MyOptionalMetricDimensionName",
    "Value": "MyOptionalMetricDimensionValue"
  }
],
"Stat": "Sum"
}
]
}
]
}
]
}
```

Weitere Informationen finden Sie [MetricDataQuery](#) in der Amazon EC2 Auto Scaling API-Referenz.

Note

Im Folgenden finden Sie einige zusätzliche Ressourcen, die Ihnen bei der Suche nach Metriknamen, Namespaces, Dimensionen und Statistiken für Metriken helfen können:
CloudWatch

- Informationen zu den verfügbaren Metriken für AWS Services finden Sie im CloudWatch Amazon-Benutzerhandbuch unter [AWS Services, die CloudWatch Metriken veröffentlichen](#).
- Den genauen Metriknamen, den Namespace und die Dimensionen (falls zutreffend) für eine CloudWatch Metrik mit dem finden Sie unter AWS CLI [list-metrics](#).

Um diese Richtlinie zu erstellen, führen Sie den [put-scaling-policy](#) Befehl mit der JSON-Datei als Eingabe aus, wie im folgenden Beispiel gezeigt.

```
aws autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \  
  --predictive-scaling-configuration file://config.json
```

Wenn der Befehl erfolgreich ausgeführt wurde, gibt er den Amazon-Ressourcennamen (ARN) der Richtlinie zurück.

```
{
```

```
"PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",
  "Alarms": []
}
```

Metrikberechnungs-Ausdrücke verwenden

Der folgende Abschnitt enthält Informationen und Beispiele für Richtlinien zur vorausschauenden Skalierung, die zeigen, wie Sie metrische Berechnungen in Ihrer Richtlinie verwenden können.

Themen

- [Metrikberechnung verstehen](#)
- [Beispiel für eine prädiktive Skalierungspolitik, die Metriken mit metrischer Mathematik kombiniert \(AWS CLI\)](#)
- [Beispiel für eine Richtlinie zur vorausschauenden Skalierung zur Verwendung in einem blue/green Bereitstellungsszenario \(AWS CLI\)](#)

Metrikberechnung verstehen

Wenn Sie lediglich vorhandene Metrikdaten aggregieren möchten, erspart Ihnen CloudWatch Metric Math den Aufwand und die Kosten für die Veröffentlichung einer weiteren Metrik in CloudWatch. Sie können jede verfügbare Metrik verwenden AWS , und Sie können auch Metriken verwenden, die Sie als Teil Ihrer Anwendungen definieren. Sie könnten zum Beispiel den Rückstand der Amazon SQS-Warteschlange pro Instance berechnen wollen. Dazu nehmen Sie die ungefähre Anzahl der Nachrichten, die für den Abruf aus der Warteschlange zur Verfügung stehen, und dividieren diese Zahl durch die laufende Kapazität der Auto-Scaling-Gruppe.

Weitere Informationen finden Sie unter [Verwenden von metrischer Mathematik](#) im CloudWatch Amazon-Benutzerhandbuch.

Wenn Sie sich für die Verwendung eines metrischen mathematischen Ausdrucks in Ihrer prädiktiven Skalierungsrichtlinie entscheiden, sollten Sie die folgenden Punkte beachten:

- Metrische Rechenoperationen verwenden die Datenpunkte der eindeutigen Kombination aus Metrikname, Namespace und Dimensionsschlüssel/Wertpaaren von Metriken.
- Sie können einen beliebigen arithmetischen Operator (+ - */^), jede statistische Funktion (wie AVG oder SUM) oder eine andere Funktion verwenden, die diese CloudWatch Funktion unterstützt.

- Sie können sowohl Metriken als auch die Ergebnisse anderer mathematischer Ausdrücke in den Formeln des mathematischen Ausdrucks verwenden.
- Ihre metrischen mathematischen Ausdrücke können aus verschiedenen Aggregationen zusammengesetzt sein. Für das endgültige Aggregationsergebnis ist es jedoch eine bewährte Methode, Average für die Skalierungsmetrik und Sum für die Lastmetrik zu verwenden.
- Alle Ausdrücke, die in einer metrischen Spezifikation verwendet werden, müssen letztendlich eine einzige Zeitreihe ergeben.

Um metrische Mathematik zu verwenden, gehen Sie wie folgt vor:

- Wählen Sie eine oder mehrere CloudWatch Metriken aus. Erstellen Sie dann den Ausdruck. Weitere Informationen finden Sie unter [Verwenden von metrischer Mathematik](#) im CloudWatch Amazon-Benutzerhandbuch.
- Stellen Sie mithilfe der CloudWatch Konsole oder der CloudWatch [GetMetricData](#)API sicher, dass der metrische mathematische Ausdruck gültig ist.

Beispiel für eine prädiktive Skalierungspolitik, die Metriken mit metrischer Mathematik kombiniert (AWS CLI)

Manchmal müssen Sie die Metrik nicht direkt angeben, sondern die Daten erst auf irgendeine Weise verarbeiten. Sie könnten zum Beispiel eine Anwendung haben, die Arbeit aus einer Amazon SQS-Warteschlange abruf, und Sie könnten die Anzahl der Objekte in der Warteschlange als Kriterium für die prädiktive Skalierung verwenden wollen. Die Anzahl der Nachrichten in der Warteschlange bestimmt nicht allein die Anzahl der Instances, die Sie benötigen. Daher ist weitere Arbeit erforderlich, um eine Metrik zu erstellen, die zur Berechnung des Rückstands pro Instance verwendet werden kann. Weitere Informationen finden Sie unter [Skalierungsrichtlinie basierend auf Amazon SQS](#).

Im Folgenden finden Sie ein Beispiel für eine prädiktive Skalierungsrichtlinie für dieses Szenario. Sie legt Skalierungs- und Auslastungsmetriken fest, die auf der Amazon SQS ApproximateNumberOfMessagesVisible-Metrik basieren, d.h. der Anzahl der Nachrichten, die für den Abruf aus der Warteschlange verfügbar sind. Es verwendet auch die Amazon EC2 Auto Scaling GroupInServiceInstances-Metrik und einen mathematischen Ausdruck, um den Rückstand pro Instance für die Skalierungsmetrik zu berechnen.

```
aws autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
```

```
--predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 100,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Label": "Get the queue size (the number of messages waiting to be
processed)",
            "Id": "queue_size",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Get the group size (the number of running instances)",
            "Id": "running_capacity",
            "MetricStat": {
              "Metric": {
                "MetricName": "GroupInServiceInstances",
                "Namespace": "AWS/AutoScaling",
                "Dimensions": [
                  {
                    "Name": "AutoScalingGroupName",
                    "Value": "my-asg"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          }
        ]
      }
    }
  ]
}
```

```

    {
      "Label": "Calculate the backlog per instance",
      "Id": "scaling_metric",
      "Expression": "queue_size / running_capacity",
      "ReturnData": true
    }
  ],
  "CustomizedLoadMetricSpecification": {
    "MetricDataQueries": [
      {
        "Id": "load_metric",
        "MetricStat": {
          "Metric": {
            "MetricName": "ApproximateNumberOfMessagesVisible",
            "Namespace": "AWS/SQS",
            "Dimensions": [
              {
                "Name": "QueueName",
                "Value": "my-queue"
              }
            ],
          },
          "Stat": "Sum"
        },
        "ReturnData": true
      }
    ]
  }
}
]
}
}

```

Das Beispiel gibt den ARN der Richtlinie zurück.

```

{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
  "Alarms": []
}

```

Beispiel für eine Richtlinie zur vorausschauenden Skalierung zur Verwendung in einem blue/green Bereitstellungsszenario ()AWS CLI

Ein Suchausdruck bietet eine erweiterte Option, mit der Sie eine Metrik aus mehreren Auto-Scaling-Gruppen abfragen und mathematische Ausdrücke auf sie anwenden können. Dies ist besonders nützlich für blue/green Bereitstellungen.

Note

Eine blau/grüne Bereitstellung ist eine Bereitstellungsmethode, bei der Sie zwei separate, aber identische Auto-Scaling-Gruppen erstellen. Nur eine der Gruppen empfängt den Produktionsverkehr. Der Benutzerverkehr wird zunächst auf die frühere ("blaue") Auto-Scaling-Gruppe geleitet, während eine neue Gruppe („grün“) zum Testen und Evaluieren einer neuen Version einer Anwendung oder eines Dienstes verwendet wird. Der Benutzerverkehr wird auf die grüne Auto-Scaling-Gruppe verlagert, nachdem eine neue Bereitstellung getestet und akzeptiert wurde. Sie können die blaue Gruppe dann löschen, nachdem die Bereitstellung erfolgreich war.

Wenn im Rahmen einer blue/green Bereitstellung neue Auto Scaling Scaling-Gruppen erstellt werden, kann der Metrikverlauf jeder Gruppe automatisch in die Predictive Scaling-Richtlinie aufgenommen werden, ohne dass Sie die Metrikspezifikationen ändern müssen. Weitere Informationen finden Sie im Compute-Blog unter [Verwenden von Richtlinien zur vorausschauenden Skalierung von EC2 Auto Scaling mit Blue/Green-Bereitstellungen](#). AWS

Die folgende Beispielrichtlinie zeigt, wie dies geschehen kann. In diesem Beispiel verwendet die Richtlinie die von Amazon EC2 ausgegebene CPUUtilization-Metrik. Es verwendet die Amazon EC2 Auto Scaling GroupInServiceInstances-Metrik und einen mathematischen Ausdruck, um den Wert der Skalierungsmetrik pro Instance zu berechnen. Sie gibt auch eine Kapazitätsmetrik an, um die GroupInServiceInstances-Metrik zu erhalten.

Der Suchausdruck findet das CPUUtilization von Instances in mehreren Auto-Scaling-Gruppen anhand der angegebenen Suchkriterien. Wenn Sie zu einem späteren Zeitpunkt eine neue Auto-Scaling-Gruppe erstellen, die denselben Suchkriterien entspricht, werden die CPUUtilization der Instances in der neuen Auto-Scaling-Gruppe automatisch einbezogen.

```
aws autoscaling put-scaling-policy --policy-name my-blue-green-predictive-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \  
  --metric-name CPUUtilization --metric-type GroupInServiceInstances
```

```

--predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 25,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 300))",
            "ReturnData": false
          },
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))",
            "ReturnData": false
          },
          {
            "Id": "weighted_average",
            "Expression": "load_sum / capacity_sum",
            "ReturnData": true
          }
        ]
      },
    },
    "CustomizedLoadMetricSpecification": {
      "MetricDataQueries": [
        {
          "Id": "load_sum",
          "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 3600))"
        }
      ]
    },
    "CustomizedCapacityMetricSpecification": {
      "MetricDataQueries": [
        {
          "Id": "capacity_sum",
          "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))"
        }
      ]
    }
  ]
}

```

```
    }  
  ]  
}
```

Das Beispiel gibt den ARN der Richtlinie zurück.

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-blue-green-predictive-scaling-policy",  
  "Alarms": []  
}
```

Überlegungen zu benutzerdefinierten Metriken in einer Richtlinie zur prädiktiven Skalierung

Wenn bei der Verwendung von benutzerdefinierten Metriken ein Problem auftritt, empfehlen wir Ihnen, wie folgt vorzugehen:

- Wenn eine Fehlermeldung angezeigt wird, lesen Sie die Nachricht und beheben Sie das gemeldete Problem, falls möglich.
- Wenn beim Versuch, einen Suchausdruck in einem blue/green Bereitstellungsszenario zu verwenden, ein Problem auftritt, stellen Sie zunächst sicher, dass Sie wissen, wie Sie einen Suchausdruck erstellen, der nach einer teilweisen Übereinstimmung statt nach einer exakten Übereinstimmung sucht. Vergewissern Sie sich außerdem, dass Ihre Abfrage nur die Auto-Scaling-Gruppen findet, in denen die betreffende Anwendung ausgeführt wird. Weitere Informationen zur Syntax von Suchausdrücken finden Sie unter [Syntax von CloudWatch Suchausdrücken](#) im CloudWatch Amazon-Benutzerhandbuch.
- Wenn Sie einen Ausdruck nicht im Voraus validiert haben, validiert ihn der [put-scaling-policy](#) Befehl, wenn Sie Ihre Skalierungsrichtlinie erstellen. Es besteht jedoch die Möglichkeit, dass dieser Befehl die genaue Ursache der erkannten Fehler nicht identifizieren kann. Um die Probleme zu beheben, beheben Sie die Fehler, die Sie als Antwort auf eine Anfrage an den [get-metric-data](#) Befehl erhalten. Sie können den Ausdruck auch von der CloudWatch Konsole aus beheben.
- Wenn Sie Ihre Load (Last)- und Capacity (Kapazitäts)-Diagramme in der Konsole betrachten, zeigt das Capacity (Kapazitäts)-Diagramm möglicherweise keine Daten an. Um sicherzustellen, dass die Diagramme vollständige Daten enthalten, stellen Sie sicher, dass Sie die Gruppenmetriken für Ihre Auto-Scaling-Gruppen konsequent aktivieren. Weitere Informationen finden Sie unter [Aktivieren der Auto-Scaling-Metriken \(Konsole\)](#).

- Die Angabe der Kapazitätsmetrik ist nur für blau/grüne Bereitstellungen sinnvoll, wenn Sie Anwendungen haben, die während ihrer Lebensdauer in verschiedenen Auto-Scaling-Gruppen laufen. Mit dieser benutzerdefinierten Metrik können Sie die Gesamtkapazität mehrerer Auto-Scaling-Gruppen angeben. Die prädiktive Skalierung nutzt dies, um historische Daten in den Capacity (Kapazitäts)-Diagrammen in der Konsole anzuzeigen.
- Sie müssen `false` für `ReturnData` angeben, wenn `MetricDataQueries` die Funktion `SEARCH()` allein ohne eine mathematische Funktion wie `SUM()` angibt. Das liegt daran, dass Suchausdrücke mehrere Zeitreihen zurückgeben können, während eine auf einem Ausdruck basierende Metrikspezifikation nur eine Zeitreihe zurückgeben kann.
- Alle an einem Suchausdruck beteiligten Metriken sollten die gleiche Auflösung haben.

Einschränkungen

- Sie können Datenpunkte von bis zu 10 Metriken in einer Metrikspezifikation abfragen.
- Für die Zwecke dieses Limits zählt ein Ausdruck als eine Metrik.

Steuern welche Auto-Scaling-Instances beim Abskalieren beendet werden

Amazon EC2 Auto Scaling verwendet Kündigungsrichtlinien, um die Reihenfolge für das Beenden von Instances festzulegen. Sie können eine vordefinierte Richtlinie verwenden oder eine benutzerdefinierte Richtlinie erstellen, um Ihre spezifischen Anforderungen zu erfüllen. Durch die Verwendung einer benutzerdefinierten Richtlinie oder Instanzskalierung als Schutz können Sie auch verhindern, dass Ihre Auto Scaling Scaling-Gruppe Instances beendet, die noch nicht bereit sind, beendet zu werden.

Inhalt

- [Wenn Amazon EC2 Auto Scaling Kündigungsrichtlinien verwendet](#)
- [Kündigungsrichtlinien für Amazon EC2 Auto Scaling konfigurieren](#)
- [Eine benutzerdefinierte Beendigungsrichtlinie mit Lambda erstellen](#)
- [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#)
- [Gestalten Sie Ihre Anwendungen so, dass sie die Instance-Kündigung ordnungsgemäß handhaben](#)

Wenn Amazon EC2 Auto Scaling Kündigungsrichtlinien verwendet

In den folgenden Abschnitten werden die Szenarien beschrieben, in denen Amazon EC2 Auto Scaling Beendigungsrichtlinien verwendet.

Inhalt

- [Ereignisse skalieren](#)
- [Instance-Aktualisierung](#)
- [Neuausgleich der Availability Zone](#)

Ereignisse skalieren

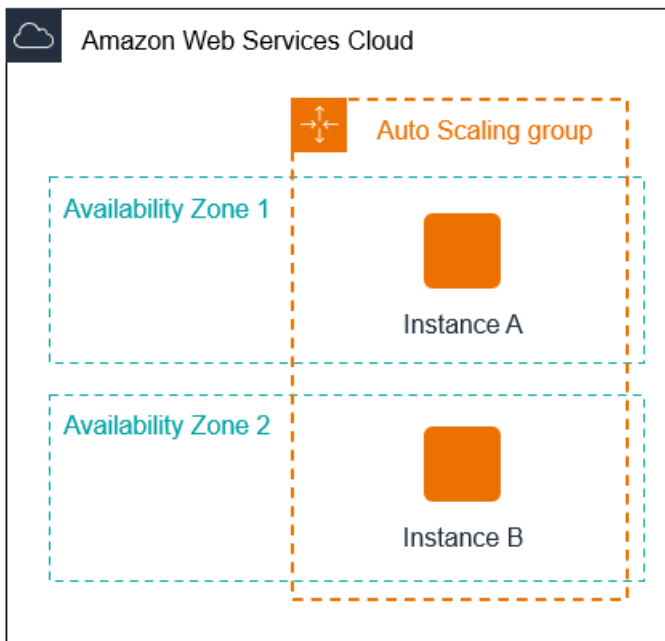
Eine Skalierung tritt ein, wenn es einen neuen Wert für die gewünschte Kapazität einer Auto Scaling Scaling-Gruppe gibt, der niedriger ist als die aktuelle Kapazität der Gruppe.

In den folgenden Szenarien kommt es zu einer Zunahme von Ereignissen:

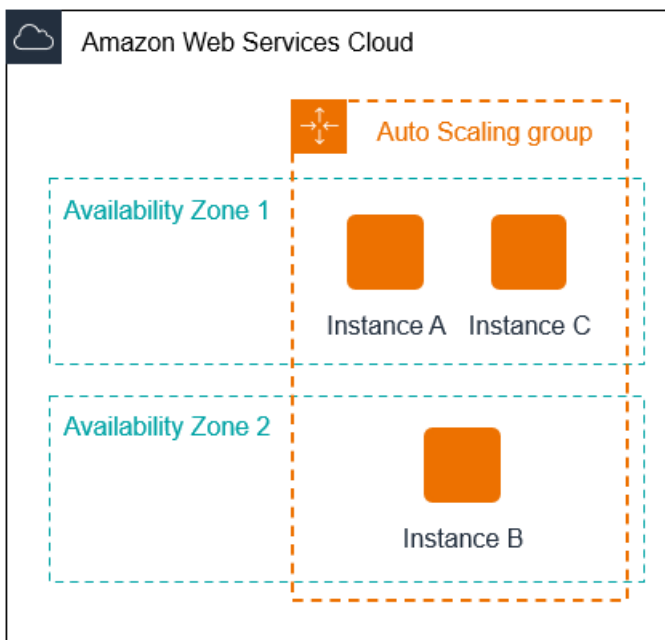
- Wenn dynamische Skalierungsrichtlinien verwendet werden, nimmt die Größe der Gruppe aufgrund von Änderungen des Werts einer Metrik ab
- Bei Verwendung der geplanten Skalierung nimmt die Größe der Gruppe infolge einer geplanten Aktion ab
- Wenn Sie die Gruppengröße manuell verkleinern

Das folgende Beispiel zeigt, wie Kündigungsrichtlinien funktionieren, wenn ein bestimmtes Ereignis eintritt.

1. Die Auto-Scaling-Gruppe in diesem Beispiel hat einen Instance-Typ, zwei Availability Zones und eine gewünschte Kapazität von zwei Instances. Es verfügt auch über eine dynamische Skalierungsrichtlinie, die Instances hinzufügt und entfernt, wenn die Ressourcenauslastung zunimmt oder abnimmt. Die zwei Instances in dieser Gruppe sind auf die zwei Availability Zones verteilt, wie im folgenden Diagramm dargestellt.

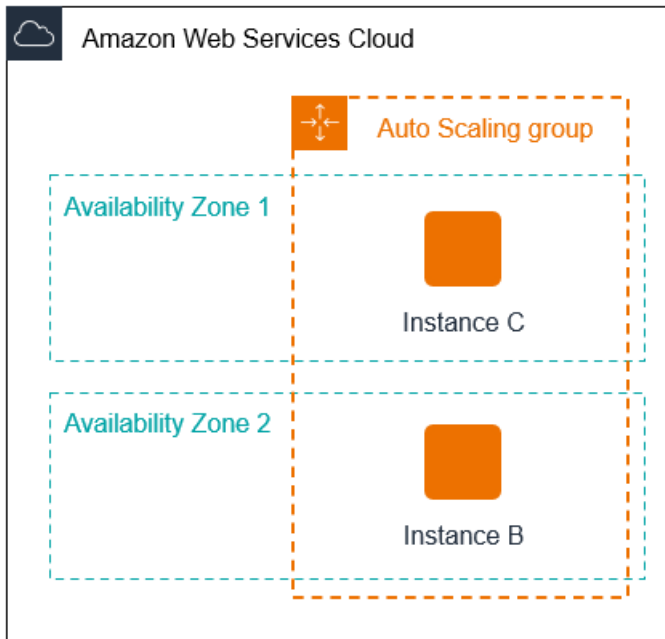


2. Wenn die Auto-Scaling-Gruppe skaliert wird, startet Amazon EC2 Auto Scaling eine neue Instance. Die Auto-Scaling-Gruppe verfügt nun über drei Instances, die auf die beiden Availability Zones verteilt sind, wie im folgenden Diagramm dargestellt.



3. Wenn die Auto-Scaling-Gruppe skaliert wird, beendet Amazon EC2 Auto Scaling eine der Instances.
4. Wenn Sie der Gruppe keine bestimmte Beendigungsrichtlinie zugewiesen haben, verwendet Amazon EC2 Auto Scaling die Standardbeendigungsrichtlinie. Sie wählt die Availability Zone mit zwei Instances aus und beendet die Instance, die über eine Startkonfiguration, eine andere

Startvorlage oder die älteste Version der aktuellen Startvorlage gestartet wurde. Wenn die Instances mit derselben Startvorlage und Version gestartet wurden, wählt Amazon EC2 Auto Scaling die Instance aus, die der nächsten Abrechnungsstunde am nächsten ist, und beendet sie.



Instance-Aktualisierung

Sie können eine Instance-Aktualisierung starten, um die Instances in Ihrer Auto Scaling Scaling-Gruppe zu aktualisieren. Während einer Instance-Aktualisierung beendet Amazon EC2 Auto Scaling Instances in der Gruppe und startet dann Ersetzungen für die beendeten Instances. Die Beendigungsrichtlinie für die Auto-Scaling-Gruppe steuert, welche Instances zuerst ersetzt werden.

Neuausgleich der Availability Zone

Amazon EC2 Auto Scaling gleicht Ihre Kapazität gleichmäßig über die Availability Zones aus, die für Ihre Auto-Scaling-Gruppe aktiviert sind. Dies trägt dazu bei, die Auswirkungen eines Ausfalls der Availability Zone zu reduzieren. Wenn die Verteilung der Kapazität über Availability Zones nicht ausgeglichen ist, gleicht Amazon EC2 Auto Scaling die Auto-Scaling-Gruppe neu aus, indem Instances in den aktivierten Availability Zones mit den wenigsten Instances gestartet und Instances an anderer Stelle beendet werden. Die Beendigungsrichtlinie steuert, welche Instances zuerst für die Beendigung priorisiert werden.

Es gibt eine Reihe von Gründen, warum die Verteilung von Instances über Availability Zones aus dem Gleichgewicht geraten kann.

Entfernen von Instances

Wenn Sie Instances von Ihrer Auto-Scaling-Gruppe trennen, setzen Sie Instances in den Standby-Modus oder beenden Instances explizit und verringern die gewünschte Kapazität, wodurch das Starten von Ersatz-Instances verhindert wird. Dadurch kann die Gruppe unausgewogen sein. Wenn dies auftritt, kann Amazon EC2 Auto Scaling dies kompensieren und das Gleichgewicht der Availability Zones wiederherstellen.

Verwenden anderer Availability Zones als ursprünglich angegeben

Wenn Sie Ihre Auto-Scaling-Gruppe erweitern, um zusätzliche Availability Zones einzuschließen oder Sie ändern, welche Availability Zones verwendet werden, startet Amazon EC2 Auto Scaling Instances in den neuen Availability Zones, und Instances in den anderen Zonen werden beendet, um sicherzustellen, dass sich Ihre Auto-Scaling-Gruppe gleichmäßig über Availability Zones erstreckt.

Ausfall der Verfügbarkeit

Verfügbarkeitsausfälle sind selten. Wenn jedoch eine Availability Zone nicht verfügbar ist und später wiederhergestellt wird, kann die Auto-Scaling-Gruppe zwischen Availability Zones unausgewogen sein. Amazon EC2 Auto Scaling versucht, die Gruppe schrittweise neu auszugleichen, und durch den Neuausgleich können Instances in anderen Zonen beendet werden.

Nehmen wir das Beispiel mit einer Auto-Scaling-Gruppe mit einem Instance-Typ, zwei Availability Zones und einer gewünschten Kapazität von zwei Instances. In einer Situation, in der eine Availability Zone fehlschlägt, startet Amazon EC2 Auto Scaling automatisch eine neue Instance in der fehlerfreien Availability Zone, um die Instance in der instabilen Availability Zone zu ersetzen. Wenn dann die instabile Availability Zone in einen fehlerfreien Zustand zurückkehrt, startet Amazon EC2 Auto Scaling automatisch eine neue Instance in dieser Zone, wodurch wiederum eine Instance in der nicht betroffenen Zone beendet wird.

Note

Beim Wiederherstellen des Gleichgewichts startet Amazon EC2 Auto Scaling vor dem Beenden der alten Instances neue, damit Leistung und Verfügbarkeit Ihrer Anwendung nicht beeinträchtigt werden.

Da Amazon EC2 Auto Scaling vor dem Beenden der alten Instances versucht, neue zu starten, kann das Wiederherstellen des Gleichgewichts beeinträchtigt und sogar gänzlich

unterbrochen werden, falls die angegebene maximale Kapazität nahezu oder gänzlich erreicht ist. Um dieses Problem zu vermeiden, kann das System beim Wiederherstellen des Gleichgewichts die angegebene maximale Kapazität einer Gruppe vorübergehend um 10 % (oder um die Marge einer Instance, je nachdem, welcher Wert größer ist) überschreiten. Dieser Wert kann nur erhöht werden, wenn die Gruppe die maximale Kapazität erreicht hat oder kurz davor ist und das Wiederherstellen des Gleichgewichts aufgrund einer vom Benutzer angeforderten Neuverteilung der Availability Zones oder zum Kompensieren von Verfügbarkeitsproblemen der Availability Zones erforderlich ist. Die Kapazität wird nur für die Dauer der Wiederherstellung des Gleichgewichts in der Gruppe erhöht.

Kündigungsrichtlinien für Amazon EC2 Auto Scaling konfigurieren

Eine Kündigungsrichtlinie legt die Kriterien fest, nach denen Amazon EC2 Auto Scaling Instances in einer bestimmten Reihenfolge beendet. Standardmäßig verwendet Amazon EC2 Auto Scaling eine Kündigungsrichtlinie, die darauf ausgelegt ist, zuerst Instances zu beenden, die veraltete Konfigurationen verwenden. Sie können die Kündigungsrichtlinie ändern, um zu kontrollieren, welche Instances am wichtigsten zuerst beendet werden müssen.

Wenn Amazon EC2 Auto Scaling Instances beendet, versucht es, das Gleichgewicht zwischen den Availability Zones aufrechtzuerhalten, die für Ihre Auto Scaling Scaling-Gruppe aktiviert sind. Die Aufrechterhaltung des zonalen Gleichgewichts hat Vorrang vor der Kündigungsrichtlinie. Wenn eine Availability Zone mehr Instances als andere hat, wendet Amazon EC2 Auto Scaling die Kündigungsrichtlinie zuerst auf die unausgeglichene Zone an. Wenn die Availability Zones ausgeglichen sind, wendet es die Kündigungsrichtlinie auf alle Zonen an.

Themen

- [So funktioniert die standardmäßige Kündigungsrichtlinie](#)
- [Standard-Beendigungsrichtlinie und Gruppe mit gemischten Instances](#)
- [Vordefinierte Kündigungsrichtlinien](#)
- [Ändern Sie die Kündigungsrichtlinie für eine Auto Scaling Scaling-Gruppe](#)

Note

Amazon EC2 Auto Scaling wendet Kündigungsrichtlinien nur auf Instances an, die von der Auto Scaling Scaling-Gruppe nicht als fehlerhaft eingestuft werden. Das hat zur Folge, dass

Instances, die durch Auto Scaling Scaling-Integritätsprüfungen als fehlerhaft markiert wurden, die Bewertung der Kündigungsrichtlinien umgehen.

Weitere Informationen finden Sie unter [Gestalten Sie Ihre Anwendungen so, dass sie die Instance-Kündigung ordnungsgemäß handhaben](#).

So funktioniert die standardmäßige Kündigungsrichtlinie

Wenn Amazon EC2 Auto Scaling eine Instance beenden muss, identifiziert es zunächst, welche Availability Zone (oder Zonen) die meisten Instances und mindestens eine Instance hat, die nicht vor einer Skalierung geschützt ist. Anschließend bewertet es ungeschützte Instances innerhalb der identifizierten Availability Zone wie folgt:

Instanzen, die veraltete Konfigurationen verwenden

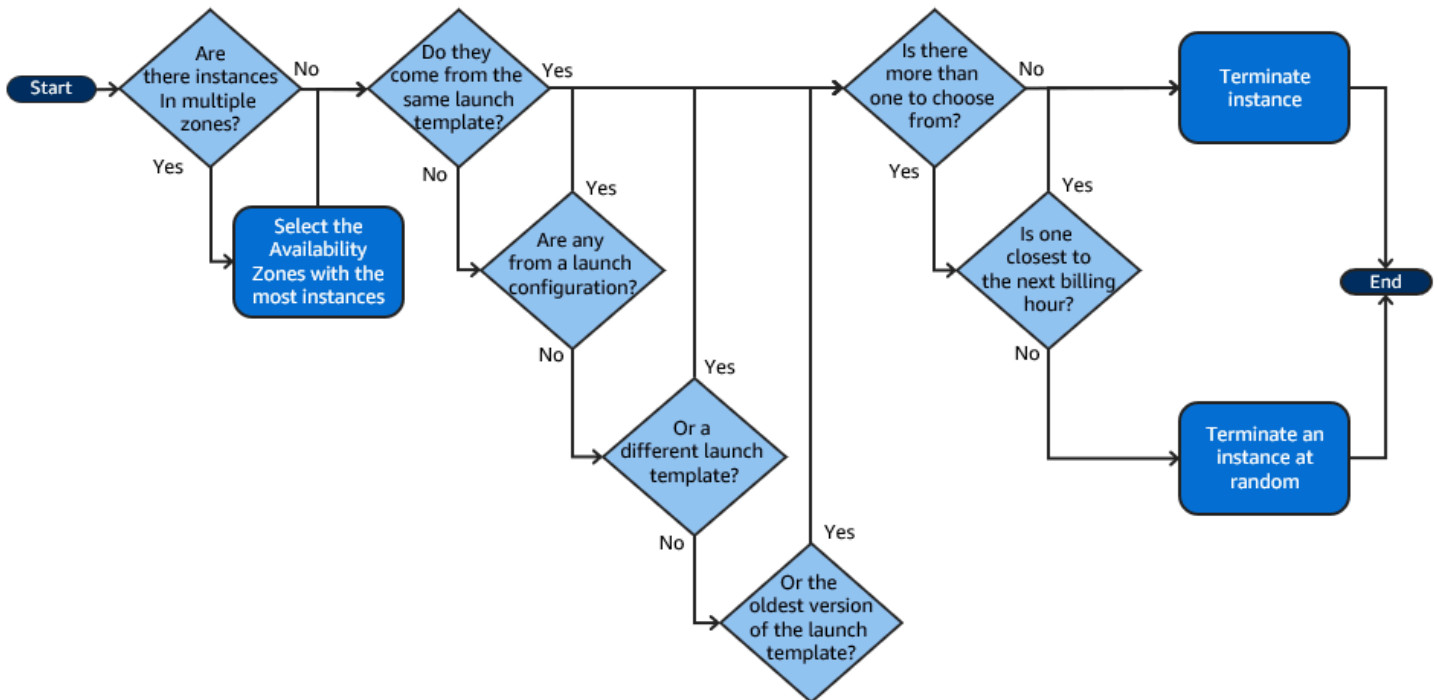
- Für Gruppen, die eine Startvorlage verwenden — Stellen Sie fest, ob eine der Instances veraltete Konfigurationen verwendet, und priorisieren Sie dabei in dieser Reihenfolge:
 1. Suchen Sie zunächst nach Instances, die mit einer Startkonfiguration gestartet wurden.
 2. Suchen Sie dann nach Instances, die mit einer anderen Startvorlage als mit der aktuellen Startvorlage gestartet wurden.
 3. Suchen Sie abschließend nach Instances, die die älteste Version der aktuellen Startvorlage verwenden.
- Für Gruppen, die eine Startkonfiguration verwenden — Stellen Sie fest, ob eine der Instances die älteste Startkonfiguration verwendet.

Wenn keine Instances mit veralteten Konfigurationen gefunden werden oder mehrere Instances zur Auswahl stehen, berücksichtigt Amazon EC2 Auto Scaling die nächsten Kriterien für Instances, die sich ihrer nächsten Abrechnungsstunde nähern.

Instances, die sich der nächsten Abrechnungsstunde nähern

Stellen Sie fest, ob eine der Instanzen, die die vorherigen Kriterien erfüllen, der nächsten Abrechnungsstunde am nächsten kommt. Wenn mehrere Instanzen gleich nah beieinander liegen, beenden Sie eine nach dem Zufallsprinzip. Auf diese Weise können Sie die Nutzung Ihrer Instances, die stündlich abgerechnet werden, maximieren. Der Großteil der EC2-Nutzung wird jetzt jedoch pro Sekunde abgerechnet, sodass diese Optimierung weniger Vorteile bietet. Weitere Informationen dazu finden Sie unter [Amazon EC2 – Preise](#).

Das folgende Flussdiagramm zeigt, wie die standardmäßige Kündigungsrichtlinie für Gruppen funktioniert, die eine Startvorlage verwenden.



Standard-Beendigungsrichtlinie und Gruppe mit gemischten Instances

Amazon EC2 Auto Scaling wendet beim Beenden von Instances in [gemischten](#) Instance-Gruppen zusätzliche Kriterien an.

Wenn Amazon EC2 Auto Scaling eine Instance beenden muss, wird zunächst anhand der Gruppeneinstellungen ermittelt, welche Kaufoption (Spot oder On-Demand) beendet werden soll. Dadurch wird sichergestellt, dass sich die Gruppe im Laufe der Zeit dem angegebenen Verhältnis von Spot- und On-Demand-Instances nähert.

Anschließend wendet sie die Kündigungsrichtlinie unabhängig innerhalb jeder Availability Zone an. Sie bestimmt, welche Spot- oder On-Demand-Instance in welcher Availability Zone beendet werden soll, um die Availability Zones im Gleichgewicht zu halten. Dieselbe Logik gilt für eine gemischte Instance-Gruppe mit definierten Gewichtungen für die Instance-Typen.

In jeder Zone funktioniert die standardmäßige Kündigungsrichtlinie wie folgt, um zu bestimmen, welche ungeschützte Instance innerhalb der identifizierten Kaufoption gekündigt werden kann:

1. Ermitteln Sie, ob eine der Instances beendet werden kann, um die Abstimmung mit der angegebenen [Zuweisungsstrategie](#) für die Auto Scaling Scaling-Gruppe zu verbessern. Wenn

keine Instanzen für die Optimierung identifiziert wurden oder mehrere Instanzen zur Auswahl stehen, wird die Bewertung fortgesetzt.

2. Stellen Sie fest, ob eine der Instanzen veraltete Konfigurationen verwendet, und priorisieren Sie dabei in dieser Reihenfolge:
 - a. Suchen Sie zunächst nach Instances, die mit einer Startkonfiguration gestartet wurden.
 - b. Suchen Sie dann nach Instances, die mit einer anderen Startvorlage als mit der aktuellen Startvorlage gestartet wurden.
 - c. Suchen Sie abschließend nach Instances, die die älteste Version der aktuellen Startvorlage verwenden.

Wenn keine Instanzen mit veralteten Konfigurationen gefunden werden oder mehrere Instanzen zur Auswahl stehen, wird die Bewertung fortgesetzt.

3. Stellen Sie fest, ob eine der Instanzen der nächsten Abrechnungsstunde am nächsten ist. Wenn mehrere Instanzen gleich nah beieinander liegen, wählen Sie nach dem Zufallsprinzip eine aus.

Vordefinierte Kündigungsrichtlinien

Sie wählen aus den folgenden vordefinierten Kündigungsrichtlinien:

- **Default**— Beenden Sie Instances gemäß der Standard-Kündigungsrichtlinie.
- **AllocationStrategy**— Beenden Sie Instances in der Auto Scaling Scaling-Gruppe, um die verbleibenden Instances an der Zuweisungsstrategie für den Instance-Typ auszurichten, der beendet wird (entweder eine Spot-Instance oder eine On-Demand-Instance). Diese Richtlinie ist nützlich, wenn Ihre bevorzugte Instance-Typen sich geändert haben. Wenn die Spot-Zuweisungsstrategie `lowest-price` lautet, können Sie Spot-Instances allmählich auf die N günstigsten Spot-Instance-Pools neu verteilen. Wenn die Spot-Zuweisungsstrategie `capacity-optimized` lautet, können Sie Spot-Instances allmählich auf mehrere Spot-Pools verteilen, in denen mehr Spot-Kapazität verfügbar ist. Sie können auch schrittweise On-Demand-Instances einer niedrigeren Priorität durch On-Demand-Instances einer höheren Priorität ersetzen.
- **OldestLaunchTemplate**— Beendet Instances mit der ältesten Startvorlage. Mit dieser Richtlinie werden Instances, die eine langfristige Startvorlage verwenden, zuerst beendet, gefolgt von Instances, die eine älteste Version der aktuellen Startvorlage verwenden. Diese Richtlinie ist nützlich, wenn Sie eine Gruppe aktualisieren und die Instances einer früheren Konfiguration auslaufen lassen.

- **OldestLaunchConfiguration**— Beendet Instances mit der ältesten Startkonfiguration. Diese Richtlinie ist nützlich, wenn Sie eine Gruppe aktualisieren und die Instances einer früheren Konfiguration auslaufen lassen. Mit dieser Richtlinie werden Instances, welche die nicht aktuelle Startkonfiguration verwenden, zuerst beendet.
- **ClosestToNextInstanceHour**— Beendet Instances, die der nächsten Abrechnungsstunde am nächsten sind. Diese Richtlinie hilft Ihnen, die Nutzung Ihrer Instances mit Stundengebühr zu maximieren.
- **NewestInstance**— Beendet die neueste Instanz in der Gruppe. Diese Richtlinie ist nützlich, wenn Sie eine neue Startkonfiguration testen, sie aber in der Produktionsumgebung nicht weiter verwenden möchten.
- **OldestInstance**— Beendet die älteste Instanz in der Gruppe. Diese Option ist nützlich, wenn Sie die Instances in der Auto-Scaling-Gruppe auf einen neuen EC2-Instance-Typ upgraden. Sie können nach und nach Instances des alten Typs durch Instances des neuen Typs ersetzen.

Note

Amazon EC2 Auto Scaling gleicht Instances immer zuerst über Availability Zones aus, unabhängig davon, welche Beendigungsrichtlinie verwendet wird. Daher können Situationen auftreten, in denen neuere Instances vor älteren Instances beendet werden. Beispielsweise, wenn eine kürzlich hinzugefügte Availability Zone vorhanden ist oder eine Availability Zone über mehr Instances verfügt als die anderen Availability Zones, die von der Gruppe verwendet werden.

Ändern Sie die Kündigungsrichtlinie für eine Auto Scaling Scaling-Gruppe

Verwenden Sie eine der folgenden Methoden, um die Kündigungsrichtlinie für Ihre Auto Scaling Scaling-Gruppe zu ändern.

Console

Sie können die Kündigungsrichtlinie nicht ändern, wenn Sie zum ersten Mal eine Auto Scaling Scaling-Gruppe in der Amazon EC2 Auto Scaling Scaling-Konsole erstellen. Die standardmäßige Beendigungsrichtlinie wird automatisch verwendet. Nachdem Ihre Auto Scaling Scaling-Gruppe erstellt wurde, können Sie die Standardrichtlinie durch eine andere Kündigungsrichtlinie oder durch mehrere Kündigungsrichtlinien ersetzen, die in der Reihenfolge aufgeführt sind, in der sie gelten sollen.

So ändern Sie die Kündigungsrichtlinie für eine Auto Scaling Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Erweiterte Konfigurationen, Bearbeiten.
4. Wählen Sie für Beendigungsrichtlinien eine oder mehrere Beendigungsrichtlinien aus. Wenn Sie mehrere Richtlinien auswählen, geben Sie diese in der Reihenfolge an, in der sie ausgewertet werden sollen.

Sie können optional die Option Benutzerdefinierte Beendigungsrichtlinie wählen und dann eine Lambda-Funktion auswählen, die Ihren Anforderungen entspricht. Wenn Sie Versionen und Aliase für Ihre Lambda-Funktion erstellt haben, können Sie eine Version oder einen Alias aus der Dropdown-Liste Version/Alias auswählen. Um die unveröffentlichte Version Ihrer Lambda-Funktion zu verwenden, lassen Sie Version/Alias auf den Standardwert eingestellt. Weitere Informationen finden Sie unter [Eine benutzerdefinierte Beendigungsrichtlinie mit Lambda erstellen](#).

Note

Wenn Sie mehrere Richtlinien verwenden, muss deren Reihenfolge korrekt festgelegt werden:

- Wenn Sie die Standardrichtlinie verwenden, muss sie die letzte Richtlinie in der Liste sein.
- Wenn Sie eine Custom termination policy (benutzerdefinierte Beendigungsrichtlinie) verwenden, muss diese die erste Richtlinie in der Liste sein.

5. Wählen Sie Aktualisieren aus.

AWS CLI

Die Standardbeendigungsrichtlinie wird automatisch verwendet, es sei denn, es wird eine andere Richtlinie angegeben.

So ändern Sie die Kündigungsrichtlinie für eine Auto Scaling Scaling-Gruppe

Verwenden Sie einen der folgenden Befehle:

- [create-auto-scaling-group](#)
- [update-auto-scaling-group](#)

Sie können diese Beendigungsrichtlinien einzeln verwenden oder sie zu einer Liste von Richtlinien zusammenführen. Nutzen Sie z. B. den folgenden Befehl, um eine Auto-Scaling-Gruppe zu aktualisieren, damit sie zuerst die Richtlinie `OldestLaunchConfiguration` und dann die Richtlinie `ClosestToNextInstanceHour` verwendet.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --  
termination-policies "OldestLaunchConfiguration" "ClosestToNextInstanceHour"
```

Falls Sie die `Default`-Beendigungsrichtlinie verwenden, setzen Sie sie ans Ende der Liste der Beendigungsrichtlinien. Beispiel, `--termination-policies "OldestLaunchConfiguration" "Default"`.

Um eine benutzerdefinierte Kündigungsrichtlinie zu verwenden, müssen Sie zunächst Ihre Kündigungsrichtlinie mithilfe von erstellen AWS Lambda. Um die Lambda-Funktion zur Verwendung als Beendigungsrichtlinie anzugeben, setzen Sie sie an die erste Stelle in der Liste der Beendigungsrichtlinien. Beispiel, `--termination-policies "arn:aws:lambda:us-west-2:123456789012:function:HelloFunction:prod" "OldestLaunchConfiguration"`. Weitere Informationen finden Sie unter [Eine benutzerdefinierte Beendigungsrichtlinie mit Lambda erstellen](#).

Eine benutzerdefinierte Beendigungsrichtlinie mit Lambda erstellen

Amazon EC2 Auto Scaling verwendet Beendigungsrichtlinien, um zu priorisieren, welche Instances zuerst beendet werden sollen, wenn die Größe Ihrer Auto-Scaling-Gruppe verringert (abskaliert) wird. Ihre Auto-Scaling-Gruppe verwendet eine Standard-Beendigungsrichtlinie, Sie können jedoch optional eigene Beendigungsrichtlinien auswählen oder erstellen. Weitere Informationen zur Auswahl einer vordefinierten Beendigungsrichtlinie finden Sie unter [Kündigungsrichtlinien für Amazon EC2 Auto Scaling konfigurieren](#).

In diesem Thema erfahren Sie, wie Sie eine benutzerdefinierte Beendigungsrichtlinie mithilfe einer AWS Lambda -Funktion erstellen, die Amazon EC2 Auto Scaling als Reaktion auf bestimmte Ereignisse aufruft. Die von Ihnen erstellte Lambda-Funktion verarbeitet die Informationen in den

Eingabedaten, die von Amazon EC2 Auto Scaling gesendet werden, und gibt eine Liste von Instances zurück, die zum Beenden bereit sind.

Eine benutzerdefinierte Beendigungsrichtlinie bietet eine bessere Kontrolle darüber, welche Instances wann beendet werden. Wenn beispielsweise Ihre Auto-Scaling-Gruppe abskaliert, kann Amazon EC2 Auto Scaling nicht ermitteln, ob Workloads ausgeführt werden, die nicht unterbrochen werden sollten. Mit einer Lambda-Funktion können Sie die Beendigungsanforderung validieren und warten, bis der Workload abgeschlossen ist, bevor Sie die Instance-ID zur Beendigung an Amazon EC2 Auto Scaling zurückgeben.

Inhalt

- [Eingabedaten](#)
- [Antwortdaten](#)
- [Überlegungen](#)
- [So erstellen Sie die Lambda-Funktion:](#)
- [Einschränkungen](#)

Eingabedaten

Amazon EC2 Auto Scaling generiert eine JSON-Nutzlast zur Skalierung von Ereignissen und tut dies auch, wenn Instances aufgrund der maximalen Instance-Lebensdauer oder der Instance-Aktualisierungsfunktionen kurz vor der Beendigung stehen. Es generiert auch eine JSON-Nutzlast für die Skalierung von Ereignissen, die es auslösen kann, wenn Ihre Gruppe zwischen Availability Zones neu verteilt wird.

Diese Nutzlast enthält Informationen über die Kapazität, die Amazon EC2 Auto Scaling beendet werden muss, eine Liste der Instances, die es für die Beendigung vorschlägt, und das Ereignis, das die Beendigung ausgelöst hat.

Es folgt ein Beispiel einer Nutzlast:

```
{
  "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:<account-
id>:autoScalingGroup:d4738357-2d40-4038-ae7e-b00ae0227003:autoScalingGroupName/my-asg",
  "AutoScalingGroupName": "my-asg",
  "CapacityToTerminate": [
    {
      "AvailabilityZone": "us-east-1b",
```

```

    "Capacity": 2,
    "InstanceMarketOption": "on-demand"
  },
  {
    "AvailabilityZone": "us-east-1b",
    "Capacity": 1,
    "InstanceMarketOption": "spot"
  },
  {
    "AvailabilityZone": "us-east-1c",
    "Capacity": 3,
    "InstanceMarketOption": "on-demand"
  }
],
"Instances": [
  {
    "AvailabilityZone": "us-east-1b",
    "InstanceId": "i-0056faf8da3e1f75d",
    "InstanceType": "t2.nano",
    "InstanceMarketOption": "on-demand"
  },
  {
    "AvailabilityZone": "us-east-1c",
    "InstanceId": "i-02e1c69383a3ed501",
    "InstanceType": "t2.nano",
    "InstanceMarketOption": "on-demand"
  },
  {
    "AvailabilityZone": "us-east-1c",
    "InstanceId": "i-036bc44b6092c01c7",
    "InstanceType": "t2.nano",
    "InstanceMarketOption": "on-demand"
  },
  ...
],
"Cause": "SCALE_IN"
}

```

Die Nutzlast enthält den Namen der Auto-Scaling-Gruppe, ihren Amazon-Ressourcennamen (ARN) und die folgenden Elemente:

- `CapacityToTerminate` beschreibt, wie viel Ihrer Spot- oder On-Demand-Kapazität in einer bestimmten Availability Zone beendet wird.

- `Instances` stellt die Instances dar, die Amazon EC2 Auto Scaling basierend auf den Informationen in `CapacityToTerminate` vorschlägt.
- `Cause` beschreibt das Ereignis, das die Beendigung ausgelöst hat: `SCALE_IN`, `INSTANCE_REFRESH`, `MAX_INSTANCE_LIFETIME` oder `REBALANCE`.

In den folgenden Informationen werden die wichtigsten Faktoren erläutert, wie Amazon EC2 Auto Scaling die Instances in den Eingabedaten generiert:

- Die Aufrechterhaltung des Gleichgewichts zwischen den Availability Zones hat Vorrang, wenn eine Instance aufgrund skaliertes Ereignisses und aufgrund von Abbrüchen aufgrund von Instanzaktualisierungen beendet wird. Wenn daher eine Availability Zone über mehr Instances verfügt als die anderen Availability Zones, die von der Gruppe verwendet werden, umfassen die Eingabedaten nur Instances zur Beendigung, die aus der unausgewogenen Availability Zone stammen. Wenn die von der Gruppe verwendeten Availability Zones ausgeglichen sind, enthalten die Eingabedaten Instances aus allen Availability Zones der Gruppe.
- Wenn Sie eine [Richtlinie für gemischte Instances](#) verwenden, hat die Aufrechterhaltung Ihrer Spot- und On-Demand-Kapazitäten auf der Grundlage Ihrer gewünschten Prozentsätze für jede Kaufoption ebenfalls Vorrang. Wir identifizieren zunächst, welcher der beiden Typen (Spot oder On-Demand) beendet werden soll. Außerdem ermitteln wir dann, welche Instances (innerhalb der identifizierten Kaufoption) in welchen Availability Zones beendet werden sollen, was dazu führt, dass die Availability Zones am stärksten ausgeglichen sind.

Antwortdaten

Die Eingabedaten und Antwortdaten arbeiten zusammen, um die Liste der zu beendenden Instances einzugrenzen.

Mit der angegebenen Eingabe sollte die Antwort Ihrer Lambda-Funktion wie im folgenden Beispiel aussehen:

```
{
  "InstanceIDs": [
    "i-02e1c69383a3ed501",
    "i-036bc44b6092c01c7",
    ...
  ]
}
```

Die InstanceIDs in der Antwort stellen die Instances dar, die zum Beenden bereit sind.

Alternativ können Sie einen anderen Satz von Instances zurückgeben, die zum Beenden bereit sind, wodurch die Instances in den Eingabedaten außer Kraft gesetzt werden. Wenn beim Aufruf Ihrer Lambda-Funktion keine Instances beendet werden können, können Sie auch keine Instances zurückgeben.

Wenn keine Instances zum Beenden bereit sind, sollte die Antwort Ihrer Lambda-Funktion wie im folgenden Beispiel aussehen:

```
{
  "InstanceIDs": [ ]
}
```

Überlegungen

Beachten Sie die folgenden Überlegungen bei der Verwendung einer benutzerdefinierten Beendigungsrichtlinie:

- Wenn Sie eine Instance zuerst in den Antwortdaten zurückgeben, wird die Beendigung nicht garantiert. Wenn beim Aufruf Ihrer Lambda-Funktion mehr als die erforderliche Anzahl von Instances zurückgegeben wird, wertet Amazon EC2 Auto Scaling jede Instance anhand der anderen Beendigungsrichtlinien aus, die Sie für Ihre Auto-Scaling-Gruppe angegeben haben. Wenn mehrere Beendigungsrichtlinien vorhanden sind, wird versucht, die nächste Beendigungsrichtlinie in der Liste anzuwenden. Wenn mehr Instances vorhanden sind, als zum Beenden erforderlich sind, wird die nächste Beendigungsrichtlinie fortgesetzt usw. Wenn keine anderen Beendigungsrichtlinien angegeben sind, wird die Standardbeendigungsrichtlinie verwendet, um zu bestimmen, welche Instances beendet werden sollen.
- Wenn keine Instances zurückgegeben werden oder Ihre Lambda-Funktion eine Zeitüberschreitung auftritt, wartet Amazon EC2 Auto Scaling kurz, bevor Sie Ihre Funktion erneut aufrufen. Bei jeder Größenordnung wird versucht, solange die gewünschte Kapazität der Gruppe geringer ist als die aktuelle Kapazität. Zum Beispiel versucht es bei Instance-Aktualisierungsbasierten Beendigungen eine Stunde lang. Wenn danach weiterhin Instances nicht beendet werden, schlägt der Instance-Aktualisierungsvorgang fehl. Bei maximaler Instance-Lebensdauer versucht Amazon EC2 Auto Scaling weiterhin, die Instance zu beenden, die als Überschreitung ihrer maximalen Lebensdauer identifiziert wird.

- Da Ihre Funktion wiederholt erneut versucht wird, stellen Sie sicher, dass Sie permanente Fehler im Code testen und beheben, bevor Sie eine Lambda-Funktion als benutzerdefinierte Beendigungsrichtlinie verwenden.
- Wenn Sie die Eingabedaten mit Ihrer eigenen Liste der zu beendenden Instances überschreiben und das Beenden dieser Instances die Availability Zones aus dem Gleichgewicht bringen, passt Amazon EC2 Auto Scaling die Kapazitätsverteilung über Availability Zones schrittweise neu aus. Zuerst ruft es Ihre Lambda-Funktion auf, um zu sehen, ob es Instances gibt, die beendet werden können, damit sie bestimmen kann, ob mit dem Neuausgleich begonnen werden soll. Wenn Instances vorhanden sind, die beendet werden können, werden zuerst neue Instances gestartet. Wenn der Start der Instances abgeschlossen ist, wird festgestellt, dass die aktuelle Kapazität Ihrer Gruppe höher als die gewünschte Kapazität ist, und leitet bei Bedarf eine Skalierung ein.
- Eine benutzerdefinierte Kündigungsrichtlinie hat keinen Einfluss auf Ihre Fähigkeit, auch Scale in Protection zu verwenden, um bestimmte Instances vor der Kündigung zu schützen. Weitere Informationen finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).
- Wenn Ihre Lambda-Funktion keine Instanzen zurückgibt, verhindert dies nicht automatisch jegliche Beendigung. Amazon EC2 Auto Scaling beendet weiterhin Instances, die bei den Auto Scaling Scaling-Zustandsprüfungen als fehlerhaft eingestuft wurden, unabhängig von den Kündigungsrichtlinien.


So erstellen Sie die Lambda-Funktion:

Erstellen Sie zunächst die Lambda-Funktion, damit Sie ihren Amazon-Ressourcennamen (ARN) in den Beendigungsrichtlinien für Ihre Auto-Scaling-Gruppe angeben können.

Erstellen einer Lambda-Funktion (Konsole)

1. Öffnen Sie die Seite [Funktionen](#) der Lambda-Konsole.
2. Wählen Sie in der Navigationsleiste oben dieselbe Region aus, die Sie beim Erstellen der Auto-Scaling-Gruppe verwendet haben.
3. Wählen Sie Funktion erstellen und Von Grund auf neu erstellen aus.
4. Geben Sie unter Basic information (Grundlegende Informationen) bei Function name (Funktionsname) den Namen für Ihre Funktion ein.
5. Wählen Sie Funktion erstellen. Sie kehren zum Code und zur Konfiguration der Funktion zurück.
6. Wenn Ihre Funktion noch in der Konsole geöffnet ist, fügen Sie unter Funktionscode Ihren Code in den Editor ein.

7. Wählen Sie Bereitstellen.
8. Optional können Sie eine veröffentlichte Version der Lambda-Funktion erstellen, indem Sie die Registerkarte Versionen und dann Eine neue Version veröffentlichen auswählen. Weitere Informationen zur Versionierung in Lambda finden Sie unter [Versionen der Lambda-Funktion](#) im AWS Lambda -Entwicklerhandbuch.
9. Wenn Sie eine Version veröffentlichen möchten, wählen Sie die Registerkarte Aliasnamen aus, wenn Sie einen Alias mit dieser Version der Lambda-Funktion verbinden möchten. Weitere Informationen zu Aliassen in Lambda finden Sie unter [Versionen der Lambda-Funktion](#) im AWS Lambda -Entwicklerhandbuch.
10. Wählen Sie als Nächstes die Registerkarte Konfiguration und dann Berechtigungen aus.
11. Scrollen Sie nach unten bis zu Ressourcenbasierte Richtlinie und wählen Sie dann Hinzufügen von Berechtigungen aus. Eine ressourcenbasierte Richtlinie wird verwendet, um dem Prinzipal, der in der Richtlinie angegeben ist, Berechtigungen zum Aufrufen Ihrer Funktion zu erteilen. In diesem Fall wird der Prinzipal die [serviceverknüpfte Amazon EC2 Auto Scaling-Rolle](#) sein, die der Auto-Scaling-Gruppe zugeordnet ist.
12. In der Richtlinienanweisung konfigurieren Sie Ihre Berechtigungen:
 - a. Wählen Sie AWS-Konto.
 - b. Für Prinzipal geben Sie den ARN der aufrufenden serviceverknüpften Rolle ein, z. B. **arn:aws:iam::<aws-account-id>:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling**.
 - c. Wählen Sie für Aktion die Option lambda:. InvokeFunction
 - d. Für Anweisungs-ID geben Sie eine eindeutige Anweisungs-ID ein, wie z. B. **AllowInvokeByAutoScaling**.
 - e. Wählen Sie Speichern.
13. Nachdem Sie diese Anweisungen befolgt haben, fahren Sie als nächsten Schritt damit fort, den ARN Ihrer Funktion in den Beendigungsrichtlinien für Ihre Auto-Scaling-Gruppe anzugeben. Weitere Informationen finden Sie unter [Ändern Sie die Kündigungsrichtlinie für eine Auto Scaling Scaling-Gruppe](#).

 Note

Beispiele, die Sie als Referenz für die Entwicklung Ihrer Lambda-Funktion verwenden können, finden Sie im [GitHub Repository](#) für Amazon EC2 Auto Scaling.

Einschränkungen

- Sie können nur eine Lambda-Funktion in den Beendigungsrichtlinien für eine Auto-Scaling-Gruppe angeben. Wenn mehrere Beendigungsrichtlinien angegeben sind, muss zuerst die Lambda-Funktion angegeben werden.
- Sie können auf Ihre Lambda-Funktion verweisen, indem Sie entweder einen unqualifizierten ARN (ohne Suffix) oder einen qualifizierten ARN verwenden, der entweder eine Version oder einen Alias als Suffix hat. Wenn ein unqualifizierter ARN verwendet wird (z. B. `function:my-function`), muss die ressourcenbasierte Richtlinie für die unveröffentlichte Version Ihrer Funktion erstellt werden. Wenn ein qualifizierter ARN verwendet wird (z. B. `function:my-function:1` oder `function:my-function:prod`), muss die ressourcenbasierte Richtlinie für die spezifische veröffentlichte Version Ihrer Funktion erstellt werden.
- Sie können einen qualifizierten ARN nicht mit dem `$LATEST`-Suffix verwenden. Wenn Sie versuchen, eine benutzerdefinierte Beendigungsrichtlinie hinzuzufügen, die sich auf einen qualifizierten ARN mit dem `$LATEST`-Suffix bezieht, führt dies zu einem Fehler.
- Die Anzahl der in den Eingabedaten angegebenen Instances ist auf 30.000 Instances begrenzt. Wenn es mehr als 30.000 Instances gibt, die beendet werden könnten, nehmen die Eingabedaten `"HasMoreInstances": true` auf, um anzugeben, dass die maximale Anzahl von Instances zurückgegeben wird.
- Die maximale Laufzeit für Ihre Lambda-Funktion beträgt zwei Sekunden (2.000 Millisekunden). Als bewährte Methode sollten Sie den Zeitüberschreitungswert Ihrer Lambda-Funktion auf der Grundlage Ihrer erwarteten Laufzeit festlegen. Lambda-Funktionen haben eine Standard-Zeitüberschreitung von drei Sekunden, dies kann jedoch verringert werden.
- Wenn Ihre Laufzeit das 2-Sekunden-Limit überschreitet, wird jede aktive Skala so lange angehalten, bis die Laufzeit unter diesen Schwellenwert fällt. Suchen Sie für Lambda-Funktionen mit durchweg längeren Laufzeiten nach einer Möglichkeit, die Laufzeit zu reduzieren, z. B. indem Sie die Ergebnisse zwischenspeichern, sodass sie bei nachfolgenden Lambda-Aufrufen abgerufen werden können.
- Kontenübergreifende Lambda-Funktionen werden nicht unterstützt. Die Lambda-Funktion, die als benutzerdefinierte Kündigungsrichtlinie verwendet wird, muss sich in derselben AWS-Konto Auto Scaling Scaling-Gruppe befinden. Die Einrichtungsanweisungen auf dieser Seite konzentrieren sich speziell auf Konfigurationen mit demselben Konto.
- Benutzerdefinierte Kündigungsrichtlinien gelten nicht für fehlerhafte Instances. Ungesunde Instances gelten immer als kündigungsfähig, und die Lambda-Funktion wertet nur die Kündigungsreihenfolge für die verbleibenden Instances aus.

Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren

Mit dem Instance Scale-In Protection haben Sie die Kontrolle darüber, welche Instances Amazon EC2 Auto Scaling beenden kann. Ein häufiger Anwendungsfall für diese Funktion ist die Skalierung containerbasierter Workloads. Weitere Informationen finden Sie unter [Gestalten Sie Ihre Anwendungen so, dass sie die Instance-Kündigung ordnungsgemäß handhaben](#).

Standardmäßig ist der Instanz-Scale-In-Schutz deaktiviert, wenn Sie eine Auto Scaling Scaling-Gruppe erstellen. Das bedeutet, dass Amazon EC2 Auto Scaling jede Instance in der Gruppe beenden kann.

Sie können Instances schützen, sobald sie gestartet werden, indem Sie die Instance-Abskalierungsschutz-Einstellung für Ihre Auto-Scaling-Gruppe aktivieren. Der Instance-Skalierungsschutz tritt in Kraft, sobald der Instance-Status `InService` lautet. Um dann zu kontrollieren, welche Instances beendet werden können, deaktivieren Sie die Abskalierungsschutz-Einstellung für einzelne Instances innerhalb der Auto-Scaling-Gruppe. Auf diese Weise können Sie bestimmte Instances weiterhin vor dem ungewollten Beenden schützen.

Topics

- [Überlegungen](#)
- [Ändern Sie den Scale-In-Schutz für eine Auto Scaling Scaling-Gruppe](#)
- [Ändern Sie den Scale-In-Schutz für eine Instanz](#)

Überlegungen

Bei der Verwendung von Instance Scale-In Protection sollten Sie Folgendes beachten:

- Wenn alle Instances in einer Auto Scaling-Gruppe vor einer Skalierung geschützt sind und ein Scale-In-Ereignis eintritt, wird die gewünschte Kapazität verringert. Die Auto Scaling Scaling-Gruppe kann die erforderliche Anzahl von Instances jedoch erst beenden, wenn ihre Instanzskalierung in den Schutzeinstellungen deaktiviert ist. In der AWS-Managementkonsole enthält der Aktivitätsverlauf für die Auto Scaling Scaling-Gruppe die folgende Meldung, wenn alle Instances in einer Auto Scaling-Gruppe vor dem Einkalieren geschützt sind, wenn ein Scale-In-Ereignis eintritt: `Could not scale to desired capacity because all remaining instances are protected from scale in.`

- Wenn Sie eine Instance trennen, die vor dem Skalieren geschützt ist, geht ihre Einstellung für den Instance Scale In-Schutz verloren. Wenn Sie die Instance erneut an die Gruppe anhängen, erbt sie die aktuelle Skalenschutzeinstellung für die Instanz der Gruppe. Wenn Amazon EC2 Auto Scaling eine neue Instance startet oder eine Instance aus einem warmen Pool in die Auto Scaling Scaling-Gruppe verschiebt, erbt die Instance die Instance-Skalierungsschutzeinstellung der Auto Scaling-Gruppe.
- Der Instance-Skalierungsschutz schützt die Auto-Scaling-Instances nicht vor Folgendem:
 - Ersetzung im Zuge von Zustandsprüfungen, falls die Instance Zustandsprüfungen nicht besteht. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).
 - Spot-Instance-Unterbrechungen Eine Spot-Instance wird beendet, wenn keine Kapazität mehr verfügbar ist oder der Spot-Preis Ihren Höchstpreis übersteigt.
 - Eine Kapazitätsblock-Reservierung endet. Amazon EC2 fordert die Capacity Block-Instances zurück, auch wenn sie vor Skalierung geschützt sind.
 - Manuelles Beenden mit dem Befehl. `terminate-instance-in-auto-scaling-group` Weitere Informationen finden Sie unter [Beenden einer Instance in Ihrer Auto-Scaling-Gruppe \(AWS CLI\)](#).
 - Manuelles Beenden über die Amazon EC2 EC2-Konsole, CLI-Befehle und API-Operationen. Aktivieren Sie den Amazon EC2-Beendigungsschutz, um Auto-Scaling-Instances vor manueller Beendigung zu schützen. (Dies verhindert nicht, dass Amazon EC2 Auto Scaling Instances beendet oder manuell über den `terminate-instance-in-auto-scaling-group` Befehl beendet.) Informationen zur Aktivierung des Amazon EC2 EC2-Kündigungsschutzes in einer Startvorlage finden Sie unter [Erstellen einer Startvorlage mithilfe erweiterter Einstellungen](#).

Ändern Sie den Scale-In-Schutz für eine Auto Scaling Scaling-Gruppe

Sie können den Instance-Skalierungsschutz für eine Auto-Scaling-Gruppe aktivieren oder deaktivieren. Wenn Sie ihn aktivieren, ist für alle neuen Instances, die von der Gruppe gestartet werden, der Instanz-Scale-In-Schutz aktiviert.

Das Aktivieren oder Deaktivieren dieser Einstellung für eine Auto Scaling Scaling-Gruppe hat keine Auswirkungen auf bestehende Instances.

Console

So aktivieren Sie den Scale-In-Schutz für eine neue Auto Scaling Scaling-Gruppe

Wenn Sie die Auto Scaling Scaling-Gruppe erstellen, aktivieren Sie auf der Seite Gruppengröße und Skalierungsrichtlinien konfigurieren unter Instance Scale-In Protection das Kontrollkästchen Instance Scale-In Protection aktivieren.

Um den Scale-in-Schutz für eine bestehende Gruppe zu aktivieren oder zu deaktivieren

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Erweiterte Konfigurationen, Bearbeiten.
4. Aktivieren oder deaktivieren Sie für Instance Scale-In Protection das Kontrollkästchen Instance-Scale Protection aktivieren oder deaktivieren, um diese Option nach Bedarf zu aktivieren oder zu deaktivieren.
5. Wählen Sie Aktualisieren aus.

AWS CLI

So aktivieren Sie den Scale-In-Schutz für eine neue Auto Scaling Scaling-Gruppe

Verwenden Sie den folgenden [create-auto-scaling-group](#)-Befehl, um den Instance-Skalierungsschutz zu aktivieren:

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --new-instances-protected-from-scale-in ...
```

Um den Scale-In-Schutz für eine bestehende Gruppe zu aktivieren

Verwenden Sie den folgenden [update-auto-scaling-group](#)-Befehl, um den Instanz-Scale-In-Schutz für die angegebene Auto Scaling Scaling-Gruppe zu aktivieren.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --new-instances-protected-from-scale-in
```

Um den Scale-In-Schutz für eine bestehende Gruppe zu deaktivieren

Verwenden Sie den folgenden Befehl, um den Instance-Skalierungsschutz für die angegebene Gruppe zu deaktivieren:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --no-new-instances-protected-from-scale-in
```

Ändern Sie den Scale-In-Schutz für eine Instanz

Standardmäßig übernehmen Instances die Instance-Skalierungsschutzeinstellung der Auto-Scaling-Gruppe, der sie angehören. Sie können den Instanz-Scale-In-Schutz jedoch für einzelne Instances nach deren Start aktivieren oder deaktivieren.

Console

Um den Scale-in-Schutz für eine Instance zu aktivieren oder zu deaktivieren

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Instance management (Instance-Verwaltung) unter Instances eine Instance aus.
4. Um den Instance-Skalierungsschutz zu aktivieren, wählen Sie Actions (Aktionen), Set Scale In Protection (Skalierungsschutz festlegen) aus. Wählen Sie nach Aufforderung Set Scale In Protection (Skalierungsschutz einrichten) aus.
5. Um den Instance-Abwärtsskalierungsschutz zu deaktivieren, wählen Sie Actions (Aktionen), Remove Scale In Protection (Skalierungsschutz entfernen) aus. Wählen Sie nach Aufforderung Remove Scale In Protection (Skalierungsschutz entfernen) aus.

AWS CLI

Um den Scale-In-Schutz für eine Instance zu aktivieren

Verwenden Sie den folgenden [set-instance-protection](#)-Befehl, um den Instance-Skalierungsschutz für die angegebene Instance zu aktivieren:

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-group-name my-asg --protected-from-scale-in
```

Um den Scale-In-Schutz für eine Instance zu deaktivieren

Verwenden Sie den folgenden Befehl, um den Instance-Skalierungsschutz der angegebenen Instance zu deaktivieren:

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-group-name my-asg --no-protected-from-scale-in
```

Note

Denken Sie daran, dass der Instance Scale-In Protection nicht garantiert, dass Instances im Falle eines menschlichen Fehlers nicht beendet werden, z. B. wenn jemand eine Instance manuell über die Amazon EC2 EC2-Konsole beendet oder. AWS CLI Wenn Sie Ihre Instance davor schützen möchten, dass sie versehentlich beendet wird, verwenden Sie den Amazon EC2-Beendigungsschutz. Selbst bei aktiviertem Beendigungsschutz und Instance-Scale-In-Schutz können Daten, die im Instance-Speicher gespeichert werden, verloren gehen, wenn eine Zustandsprüfung feststellt, dass eine Instance fehlerhaft ist oder wenn die Gruppe selbst versehentlich gelöscht wurde. Wie bei jeder Umgebung ist es eine bewährte Vorgehensweise, Ihre Daten häufig zu sichern bzw. zu für Ihre Business Continuity-Anforderungen geeigneten Intervallen.

Gestalten Sie Ihre Anwendungen so, dass sie die Instance-Kündigung ordnungsgemäß handhaben

In diesem Thema werden Funktionen beschrieben, mit denen Sie verhindern können, dass Ihre Amazon EC2 Auto Scaling Scaling-Gruppe Amazon EC2-Instances beendet, die noch nicht zum Beenden bereit sind. Standardmäßig hat Auto Scaling keinen Einblick in die Anwendungen, die auf Ihren Instances ausgeführt werden. Es kann Instances beenden, bevor Ihre Anwendung ordnungsgemäß heruntergefahren oder die zugewiesenen Jobs abgeschlossen werden kann. Diese Funktionen geben Ihrer Anwendung Zeit, um laufende Arbeiten abzuschließen, den Status zu übertragen oder eine Bereinigung durchzuführen, bevor die Instance beendet wird. Sie können sie je nach den Anforderungen Ihrer Anwendung einzeln oder in Kombination verwenden.

Diese Funktionen sind besonders nützlich für statusbehaftete Workloads, bei denen jede Instance in Ihrer Flotte andere Daten, Jobs oder Status enthält als andere Instances. Das Beenden von Stateful-Instances ohne ordnungsgemäßes Herunterfahren könnte dazu führen, dass Jobs mit

langer Laufzeit von vorne neu gestartet werden, die Datenredundanz oder der Datenverlust reduziert und laufende Transaktionen oder Berechnungen unterbrochen werden. Um eine statusbehaftete Instance ordnungsgemäß herunterzufahren, sollte ihre Arbeitslast entweder entladen (Abschluss aller aktuell zugewiesenen Jobs) oder übertragen (Jobs, Daten oder Konfiguration auf eine andere aktive Instance verschieben).

Inhalt

- [Beendigungs-Lebenszyklus-Hooks](#)
- [Instance-Abskalierungsschutz](#)
- [Benutzerdefinierte Beendigungsrichtlinie](#)
- [Richtlinie für den Instanzlebenszyklus](#)
- [Kündigungen insgesamt aussetzen](#)
- [Einschränkungen](#)
- [Beispielszenarien](#)

Beendigungs-Lebenszyklus-Hooks

Ein Termination-Lifecycle-Hook verlängert die Lebensdauer Ihrer Amazon EC2 EC2-Instance, die bereits für die Kündigung ausgewählt wurde. Er bietet zusätzliche Zeit, um laufende Arbeiten abzuschließen, die derzeit der Instance zugewiesen sind, oder um den Fortschritt zu speichern und die Arbeit auf eine andere Instance zu übertragen.

Bei vielen Workloads kann ein Terminierungs-Lifecycle-Hook ausreichen, um eine Anwendung auf einer Instance, die für die Kündigung ausgewählt wurde, ordnungsgemäß herunterzufahren. Dies ist ein Best-Effort-Ansatz und kann nicht verwendet werden, um eine Kündigung zu verhindern, wenn eine Aktion im Lebenszyklus einer Kündigung abgebrochen wird. Wenn Ihr Workload eine geringe Toleranz gegenüber Ausfällen bei Aktionen zur Kündigung des Lebenszyklus aufweist, konfigurieren Sie [Richtlinien für den Instanzlebenszyklus](#) in Kombination mit Ihren Terminierungs-Lifecycle-Hooks, um Instances beizubehalten.

Um einen Termination-Lifecycle-Hook verwenden zu können, müssen Sie wissen, wann eine Instance für die Kündigung ausgewählt wurde. Sie haben zwei Möglichkeiten, dies herauszufinden:

Option	Description	Am besten verwendet für	Link zur Dokumentation
Innerhalb der Instance	Der Instance Metadata Service (IMDS) ist ein sicherer Endpunkt, bei dem Sie direkt von der Instance aus den Status einer Instance abfragen können. Wenn die Metadaten mit <code>Terminate</code> zurückgegeben werden, ist die Beendigung Ihrer Instance geplant.	Anwendungen, bei denen Sie eine Aktion auf der Instance ausführen müssen, bevor die Instance beendet wird.	Rufen Sie den Ziellebenszyklusstatus ab
Außerhalb der Instance	Wenn eine Instance beendet wird, wird eine Ereignisbenachrichtigung generiert. Sie können Regeln mithilfe von Amazon EventBridge, Amazon SQS, Amazon SNS oder AWS Lambda zur Erfassung dieser Ereignisse erstellen und eine Antwort aufrufen, z. B. mit einer Lambda-Funktion.	Anwendungen, die außerhalb der Instance Aktionen durchführen müssen.	Konfigurieren Sie ein Benachrichtigungsziel

Um einen Lebenszyklus-Hook verwenden zu können, müssen Sie auch wissen, wann eine Instance für die vollständige Beendigung bereit ist. Amazon EC2 Auto Scaling beendet die Instance erst, wenn sie einen [CompleteLifecycleAction](#) API-Aufruf erhält oder das Timeout abgelaufen ist, je nachdem, was zuerst eintritt.

Standardmäßig kann eine Instance aufgrund eines Beendigungs-Lebenszyklus-Hook eine Stunde lang weiterlaufen (Heartbeat-Timeout). Sie können den standardmäßigen Timeout konfigurieren, falls eine Stunde nicht ausreicht, um die Lebenszyklus-Aktion abzuschließen. Wenn eine Lebenszyklusaktion ausgeführt wird, können Sie das Timeout mit API-Aufrufen verlängern.

[RecordLifecycleActionHeartbeat](#)

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).

Instance-Abskalierungsschutz

Mit dem Instance Scale-In Protection können Sie kontrollieren, welche Instances bei Scale-In-Ereignissen für die Kündigung ausgewählt werden. Insbesondere können Sie verhindern, dass eine Instance, die aktiv einen Job mit langer Laufzeit verarbeitet, beendet wird. Wenn Sie beispielsweise containerisierte Workloads ausführen, ist es üblich, alle Instanzen zu schützen und den Schutz nur für Instanzen aufzuheben, für die es keine aktuellen oder geplanten Aufgaben gibt. Instances können weiterhin nach neuen Jobs suchen und den Schutz wieder aktivieren, wenn neue Jobs zugewiesen werden.

Sie können den Scale-In-Schutz auf Auto Scaling Scaling-Gruppenebene und Instanzebene aktivieren. Wenn Sie den Scale-in-Schutz auf Auto Scaling Scaling-Gruppenebene aktivieren, werden nur neue Instances geschützt, wenn sie erstellt werden. Für bestehende Instances können Sie den Schutz einzeln aktivieren.

Anwendungen können den Schutz entweder von den Instanzen selbst aus oder von einer zentralen Kontrollebene aus einrichten, die verwaltet, ob jede Instanz beendet werden kann. Wir empfehlen den zentralisierten Ansatz für große Flotten oder wenn der Schutz häufig umgeschaltet werden muss, da Sie so Aufrufe stapelweise aufrufen [SetInstanceProtection](#) und Probleme mit der API-Drosselung vermeiden können.

Weitere Informationen finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).

Benutzerdefinierte Beendigungsrichtlinie

Wie beim Instance Scale-In Protection hilft Ihnen eine benutzerdefinierte Kündigungsrichtlinie dabei, zu verhindern, dass Ihre Amazon EC2 Auto Scaling Scaling-Gruppe bestimmte EC2-Instances beendet. Fehlerhafte Instances können unabhängig von Ihrer benutzerdefinierten Kündigungsrichtlinie trotzdem beendet werden.

Ihre Auto Scaling Scaling-Gruppe verwendet eine Standard-Kündigungsrichtlinie, um zu bestimmen, welche Amazon EC2 EC2-Instances sie zuerst beendet. Wenn Sie mehr Kontrolle darüber haben möchten, welche Instances zuerst beendet werden, können Sie mithilfe einer Lambda-Funktion eine benutzerdefinierte Kündigungsrichtlinie implementieren. Auto Scaling ruft diese Funktion immer dann auf, wenn es eine Instanz zum Beenden auswählen muss, und beendet nur Instances, die die Funktion zurückgibt. Wenn bei der Funktion ein Fehler auftritt, ein Timeout auftritt oder eine

leere Liste zurückgegeben wird, beendet Auto Scaling keine Instances, es sei denn, die Instance ist fehlerhaft.

Eine benutzerdefinierte Kündigungsrichtlinie ist nützlich, wenn Ihre Anwendung erkennen kann, welche Instances inaktiv sind oder welche sicher beendet werden können. Dies erfordert in der Regel eine Kontrollebene, die die Arbeitslast in der gesamten Gruppe verfolgt.

Weitere Informationen finden Sie unter [Eine benutzerdefinierte Beendigungsrichtlinie mit Lambda erstellen](#).

Richtlinie für den Instanzlebenszyklus

Richtlinien für den Instance-Lebenszyklus bieten Schutz vor der Kündigung von Amazon EC2 Auto Scaling, wenn eine Aktion zum Beenden des Lebenszyklus abgebrochen wird. Im Gegensatz zu Lifecycle-Hooks allein sollen Richtlinien für den Instance-Lebenszyklus sicherstellen, dass Instances in einen beibehaltenen Zustand übergehen, wenn das ordnungsgemäße Herunterfahren nicht erfolgreich abgeschlossen werden kann.

Wenn Auto Scaling eine Instance zum Beenden auswählt, werden Ihre konfigurierten Terminierungs-Lifecycle-Hooks aufgerufen und Ihre Anwendung beginnt mit ordnungsgemäßem Herunterfahren. Wenn die Aktionen im Lebenszyklus der Kündigung erfolgreich abgeschlossen werden, wird die Instance normal beendet. Wenn jedoch eine Aktion zum Beenden des Lebenszyklus aus irgendeinem Grund abgebrochen wird, versetzt die Instance-Lebenszyklus-Richtlinie die Instance in einen beibehaltenen Zustand, anstatt sie zu beenden. Beibehaltene Instances werden nicht auf die gewünschte Kapazität Ihrer Auto Scaling Scaling-Gruppe angerechnet, sodass Ersatz-Instances automatisch gestartet werden. Es fallen Standardgebühren von Amazon EC2 sowohl für die beibehaltene Instance als auch für ihren Ersatz an, bis Sie die beibehaltene Instance manuell mithilfe der [TerminateInstancesInAutoScalingGroup](#) API beenden.

Um diese Funktion nutzen zu können, müssen Sie sowohl eine Instance-Lebenszyklus-Richtlinie mit aktiviertem `TerminateHookAbandon` Aufbewahrungs-Trigger als auch mindestens einen Terminierungs-Lifecycle-Hook konfigurieren. Da für zurückbehaltene Instances laufende Amazon EC2 EC2-Kosten anfallen und manuelle Maßnahmen erforderlich sind, ist die Überwachung von entscheidender Bedeutung. Sie sollten CloudWatch Metriken wie `GroupTerminatingRetainedInstances` aktivieren und CloudWatch Alarme einrichten, um Sie zu benachrichtigen, wenn Instances in den Status „Beibehalten“ wechseln.

Weitere Informationen finden Sie unter [Steuern Sie die Aufbewahrung von Instances mit Richtlinien für den Instance-Lebenszyklus](#).

Kündigungen insgesamt aussetzen

Wenn Sie die vollständige Kontrolle über alle EC2-Instance-Kündigungen innerhalb Ihrer Amazon EC2 Auto Scaling Scaling-Gruppe benötigen, unterbrechen Sie den Vorgang. `Terminate` Wir empfehlen, diese Option nur zu verwenden, wenn Ihnen die oben genannten Optionen nicht die Kontrolle bieten, die Sie für Ihren Service benötigen. Indem Sie aufrufen, [SuspendProcesses](#) um den `Terminate` Prozess zu unterbrechen, verhindern Sie, dass Auto Scaling versucht, aus irgendeinem Grund zu beenden, es sei denn, es handelt sich um solche, die durch eine Benutzeranfrage an die [TerminateInstancesInAutoScalingGroup](#) API ausgelöst wurden.

Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).

Einschränkungen

Important

Wenn Sie Ihre Anwendung auf Amazon EC2 Auto Scaling so entwerfen, dass sie Instance-Kündigungen ordnungsgemäß verarbeitet, sollten Sie die folgenden Einschränkungen berücksichtigen.

Ungesunde Instances umgehen einige Schutzmaßnahmen

Wenn eine Instance fehlerhaft ist, beginnt Amazon EC2 Auto Scaling damit, sie zu beenden, auch wenn Sie über benutzerdefinierte Kündigungsrichtlinien oder einen Scale-In-Schutz verfügen. Die einzige Möglichkeit, zu verhindern, dass fehlerhafte Instances durch Auto Scaling ersetzt werden, besteht darin `HealthCheck`, den `ReplaceUnhealthy`, `Terminate` or-Prozess auszusetzen. Sie können Lifecycle-Hooks und eine Instance-Lifecycle-Richtlinie verwenden, um es der Anwendung zu ermöglichen, ordnungsgemäß herunterzufahren oder alle Daten zu kopieren, die Sie wiederherstellen müssen, bevor die fehlerhafte Instance beendet wird.

Feature	Steuert fehlerfreie Instanzen	Steuert fehlerhafte Instanzen
Benutzerdefinierte Kündigungsrichtlinien	Ja	Nein

Feature	Steuert fehlerfreie Instanzen	Steuert fehlerhafte Instanzen
Abskalierungsschutz	Ja	Nein
HealthCheck ReplaceUnhealthy Aussetzen oder Terminate verarbeiten	Ja	Ja
Lebenszyklus-Hooks	Ja	Ja
Richtlinie für den Instanzlebenszyklus	Ja	Ja

Lifecycle-Hooks allein garantieren noch kein ordnungsgemäßes Herunterfahren

Standardmäßig werden Termination-Lifecycle-Hooks nach bestem Wissen und Gewissen ausgeführt. Wenn eine Aktion zum Beenden des Lebenszyklus abgebrochen wird, fährt Amazon EC2 Auto Scaling mit der sofortigen Beendigung der Instance fort. Sie können Terminierungs-Lifecycle-Hooks mit einer Instance-Lifecycle-Richtlinie kombinieren, um Instances beizubehalten, wenn die Aktionen zur Kündigung des Lebenszyklus abgebrochen werden. Mit dieser Kombination:

- Ihre Termination Lifecycle-Hooks versuchen, Ihre Anwendung ordnungsgemäß herunterzufahren, nachdem Auto Scaling eine Instance-Kündigung ausgelöst hat und die Entladung aller konfigurierten Elastic Load Balancing Balancing-Load Balancer abgeschlossen ist.
- Wenn eine Aktion zum Beenden des Lebenszyklus aus irgendeinem Grund abgebrochen wird, wechselt die Instance in den Status „Beibehalten“, anstatt beendet zu werden.
- Die beibehaltene Instance verbleibt in ihrem aktuellen Amazon EC2 EC2-Status, sodass Sie Ihre Shutdown-Verfahren manuell abschließen oder den Fehler untersuchen können.
- Sie können die gespeicherten Instances manuell beenden, indem Sie die [TerminateInstanceInAutoScalingGroup](#) API aufrufen, nachdem Sie die erforderlichen Aktionen abgeschlossen haben.

Weitere Informationen finden Sie unter [Steuern Sie die Aufbewahrung von Instances mit Richtlinien für den Instance-Lebenszyklus](#).

Bestimmte Optionen im Instance-Markt können mit begrenzter Vorankündigung unterbrochen werden

Wenn Sie Instance-Marktoptionen wie Spot-Instances und unterbrechbare Kapazitätsreservierungen in Ihrer Auto Scaling Scaling-Gruppe verwenden, kann Amazon EC2 Ihre Instances jederzeit unterbrechen und zurückfordern. Diese Unterbrechungen umgehen alle Amazon EC2 Auto Scaling Scaling-Schutzmechanismen, einschließlich:

- Beendigungs-Lebenszyklus-Hooks
- Instance-Abskalierungsschutz
- Benutzerdefinierte Kündigungsrichtlinien
- Richtlinien für den Lebenszyklus von Instanzen
- Ausgesetzte Prozesse

Wenn eine Spot-Instance eine Benachrichtigung über eine Unterbrechung erhält, haben Sie etwa zwei Minuten Zeit, um Aufgaben zum ordnungsgemäßen Herunterfahren durchzuführen. Sie können zwar Termination Lifecycle-Hooks verwenden, um auf Unterbrechungen der Spot-Instance zu reagieren, aber die Instance wird am Ende des zweiminütigen Zeitfensters zwangsweise beendet, auch wenn der Lifecycle-Hook noch läuft. Richtlinien für den Instance-Lebenszyklus können auch Unterbrechungen von Spot-Instances nicht verhindern.

Weitere Informationen zum Umgang mit Spot-Instance-Unterbrechungen finden Sie unter [Spot-Instance-Unterbrechungen](#) und [Best Practices für Amazon EC2 Spot](#) im Amazon EC2 EC2-Benutzerhandbuch.

Direkte Amazon EC2 EC2-Kündigungen umgehen alle Schutzmaßnahmen

Wenn Sie eine Instance in Ihrer Auto Scaling Scaling-Gruppe direkt mit der Amazon [TerminateInstances](#) EC2-API beenden, umgeht die Kündigung alle Amazon EC2 Auto Scaling-Schutzmechanismen.

Verwenden Sie stattdessen die [TerminateInstanceInAutoScalingGroup](#) API, um Instances in Ihrer Auto Scaling Scaling-Gruppe zu beenden und dabei Ihre konfigurierten Schutzmaßnahmen zu respektieren.

Beispielszenarien

Wenn Sie Amazon EC2 Auto Scaling verwenden, können Sie wählen, wie viel Flottenmanagement Auto Scaling in Ihrem Namen übernimmt und wie viel direkte Kontrolle Sie über Entscheidungen

zur Kündigung von EC2-Instances behalten. Je sensibler Ihre Arbeitslast auf Instance-Kündigungen reagiert, desto mehr Kontrolle möchten Sie möglicherweise behalten. In den folgenden Beispielen werden Workloads mit unterschiedlichen Toleranzstufen und die empfohlenen Konfigurationen beschrieben:

Beispiel 1: Verteilte Datenbankknoten (niedrige Toleranz)

Sie führen eine verteilte Datenbank aus, in der jede EC2-Instance eine Partition Ihrer Daten mit einem Replikationsfaktor von 3 enthält. Der Verlust mehrerer Instances, die Replikate derselben Partition enthalten, kann zu Datenverlust führen oder dazu führen, dass diese Partition nicht verfügbar ist.

Herausforderung: Auto Scaling beendet Instances möglicherweise schneller, als Daten erneut auf andere Knoten repliziert werden können, und Terminierungen könnten Ihre Kapazität unter das reduzieren, was zur Aufrechterhaltung Ihres Replikationsfaktors erforderlich ist.

Ziehen Sie die folgenden Konfigurationen in Betracht:

- Aktivieren Sie den [Instanz-Scale-In-Schutz](#) für alle Datenbankinstanzen. Entfernen Sie ihn programmgesteuert erst, nachdem Sie sich vergewissert haben, dass die Daten sicher an anderer Stelle repliziert wurden.
- Konfigurieren Sie [Terminierungs-Lifecycle-Hooks](#) mit verlängerten Timeouts in Kombination mit einer Instance-Lebenszyklus-Richtlinie, damit die Datenübertragung abgeschlossen werden kann.
- Legen Sie eine [Richtlinie für die Instanzwartung](#) fest, die einen Mindestwert von 100% vorsieht, um die erforderliche Kapazität aufrechtzuerhalten.

Beispiel 2: Auftragsverarbeitung mit langer Laufzeit (mittlere Toleranz)

Sie haben eine Amazon SQS SQS-Warteschlange, die eingehende Nachrichten für Aufträge mit langer Laufzeit sammelt. Wenn eine neue Nachricht eingeht, ruft eine EC2-Instance die Nachricht ab und startet einen Job, dessen Verarbeitung 3 Stunden dauert. Wenn die Warteschlange wächst, fügt Auto Scaling Instances auf der Grundlage Ihrer Skalierungsrichtlinien hinzu. Wenn die Warteschlange schrumpft, beendet Auto Scaling die Instances.

Herausforderung: Auto Scaling beendet möglicherweise eine Instance, die 3 Stunden nach der Verarbeitung eines Jobs hinter sich hat, und nicht eine Instance im Leerlauf. Der Job kann auf einer anderen Instance neu gestartet werden, aber Sie verlieren erhebliche Fortschritte.

Betrachten Sie die folgenden Konfigurationen:

- Konfigurieren Sie eine [benutzerdefinierte Kündigungsrichtlinie](#), bei der das Beenden inaktiver Instanzen zuerst priorisiert wird.
- Verwenden Sie [Terminierungs-Lifecycle-Hooks](#), damit laufende Jobs abgeschlossen werden können.
- Aktivieren Sie den [Instance Scale-In Protection](#) programmgesteuert, wenn eine Instance einen Job startet, und entfernen Sie ihn, wenn der Job abgeschlossen ist.

Beispiel 3: Mitarbeiterflotte für Testumgebungen (hohe Toleranz)

Sie führen eine Flotte von EC2-Instances aus, die automatisierte Tests, CI/CD Pipeline-Jobs oder Entwicklungs-Workloads ausführen. Diese Worker-Instances rufen Aufgaben aus einer Warteschlange ab, und die Testergebnisse können neu generiert werden, wenn ein Job fehlschlägt.

Herausforderung: Testaufträge können bei Scale-In-Ereignissen unterbrochen werden, aber da Tests ohne Auswirkungen wiederholt werden können, sollten Sie bei der Optimierung eher auf Kosten und Einfachheit achten als auf eine unterbrechungsfreie Verfügbarkeit.

Betrachten Sie die folgenden Konfigurationen:

- Verwenden Sie [Termination Lifecycle-Hooks](#), damit Anfragen während der Bearbeitung abgeschlossen werden können.
- Erwägen Sie die Verwendung von [Spot-Instances](#) mit einer kapazitätsoptimierten Zuweisungsstrategie, um die Kosten weiter zu senken.

Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen

In diesem Thema wird beschrieben, wie Sie einen oder mehrere Prozesse für Ihre Auto Scaling Scaling-Gruppe aussetzen und dann wieder aufnehmen, um bestimmte Vorgänge vorübergehend zu deaktivieren.

Das Aussetzen von Prozessen kann nützlich sein, wenn Sie ein Problem untersuchen oder beheben müssen, ohne dass es zu Störungen durch Skalierungsrichtlinien oder geplante Aktionen kommt. Es verhindert auch, dass Amazon EC2 Auto Scaling Instances als fehlerhaft markiert und ersetzt, während Sie Änderungen an Ihrer Auto Scaling-Gruppe vornehmen.

Themen

- [Arten von Prozessen](#)
- [Überlegungen zum Aussetzen von Prozessen](#)
- [Prozess anhalten](#)
- [Prozesse fortsetzen](#)
- [Wie sich unterbrochene Prozesse auf andere Prozesse auswirken](#)

Note

Zusätzlich zu den Unterbrechungen, die Sie initiieren, kann Amazon EC2 Auto Scaling auch Prozesse für Auto-Scaling-Gruppen unterbrechen, für die wiederholt keine Instances gestartet werden können. Dies wird als administrative Unterbrechung bezeichnet. Eine administrative Unterbrechung wird meistens für Auto-Scaling-Gruppen verwendet, die seit über 24 Stunden ohne Erfolg versuchen, Instances zu starten. Sie können Prozesse fortsetzen, die aus administrativen Gründen von Amazon EC2 Auto Scaling unterbrochen wurden.

Arten von Prozessen

Die Anhalten-Fortsetzen-Funktion unterstützt die folgenden Prozesse:

- **Launch**— Fügt Instances zur Auto Scaling-Gruppe hinzu, wenn die Gruppe horizontal skaliert wird oder wenn Amazon EC2 Auto Scaling Instances aus anderen Gründen startet, z. B. wenn Instances zu einem warmen Pool hinzugefügt werden.
- **Terminate**— Entfernt Instances aus der Auto Scaling-Gruppe, wenn die Gruppe skaliert wird oder wenn Amazon EC2 Auto Scaling beschließt, Instances aus anderen Gründen zu beenden, z. B. wenn eine Instance wegen Überschreitung ihrer maximalen Lebensdauer beendet wird oder wenn eine Zustandsprüfung nicht bestanden hat.
- **AddToLoadBalancer**— Fügt Instances der angehängten Load Balancer-Zielgruppe oder dem Classic Load Balancer hinzu, wenn sie gestartet werden. Weitere Informationen finden Sie unter [Verwenden Sie Elastic Load Balancing, um den eingehenden Anwendungsdatenverkehr in Ihrer Auto Scaling Scaling-Gruppe zu verteilen](#).
- **AlarmNotification**— Akzeptiert Benachrichtigungen von CloudWatch Alarmen, die mit dynamischen Skalierungsrichtlinien verknüpft sind. Weitere Informationen finden Sie unter [Dynamische Skalierung für Amazon EC2 Auto Scaling](#).

- **AZRebalance**— Verteilt die Anzahl der EC2-Instances in der Gruppe gleichmäßig auf alle angegebenen Availability Zones, wenn die Gruppe aus dem Gleichgewicht gerät, z. B. wenn eine zuvor nicht verfügbare Availability Zone wieder in einen fehlerfreien Zustand zurückkehrt. Weitere Informationen finden Sie unter [Wiederherstellen des Gleichgewichts von Aktivitäten](#).
- **HealthCheck**— Überprüft den Zustand der Instances und markiert eine Instance als fehlerhaft, wenn Amazon EC2 oder Elastic Load Balancing Amazon EC2 Auto Scaling mitteilt, dass die Instance fehlerhaft ist. Dieser Vorgang kann den manuell festgelegten Zustand einer Instance außer Kraft setzen. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).
- **InstanceRefresh**— Beendet und ersetzt Instances mithilfe der Instance-Aktualisierungsfunktion. Weitere Informationen finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).
- **ReplaceUnhealthy**— Beendet Instanzen, die als fehlerhaft markiert sind, und erstellt dann neue Instanzen, um sie zu ersetzen. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).
- **ScheduledActions**— Führt die geplanten Skalierungsaktionen aus, die Sie erstellen oder die für Sie erstellt werden, wenn Sie einen AWS Auto Scaling Skalierungsplan erstellen und die prädiktive Skalierung aktivieren. Weitere Informationen finden Sie unter [Geplante Skalierung für Amazon EC2 Auto Scaling](#).

Überlegungen zum Aussetzen von Prozessen

Berücksichtigen Sie Folgendes, bevor Sie Prozesse anhalten:

- Durch das Aussetzen **AlarmNotification** können Sie die Zielverfolgungs-, Step- und Simple Scaling-Richtlinien der Gruppe vorübergehend beenden, ohne die Skalierungsrichtlinien oder die zugehörigen CloudWatch Alarme zu löschen. Informationen zum vorübergehenden Beenden einzelner Skalierungsrichtlinien finden Sie unter [Eine Skalierungsrichtlinie für eine Auto-Scaling-Gruppe deaktivieren](#).
- Sie können sich dafür entscheiden, die **HealthCheck ReplaceUnhealthy** Prozesse zum Neustarten von Instances auszusetzen, ohne dass Amazon EC2 Auto Scaling die Instances aufgrund seiner Zustandsprüfungen beendet. Wenn Sie jedoch Amazon EC2 Auto Scaling benötigen, um weiterhin Zustandsprüfungen für die verbleibenden Instances durchzuführen, verwenden Sie stattdessen die Standby-Funktion. Weitere Informationen finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#).

- Wenn Sie die Prozesse `Launch` und `Terminate` oder `AZRebalance` aussetzen und dann Änderungen an Ihrer Auto-Scaling-Gruppe vornehmen, zum Beispiel indem Sie Instances abziehen oder die angegebenen Availability Zones ändern, kann Ihre Gruppe zwischen den Availability Zones unausgewogen werden. Wenn das vorkommt, verteilt Amazon EC2 Auto Scaling nach dem Fortsetzen der angehaltenen Prozesse die Instances schrittweise gleichmäßig auf die Availability Zones.
- Wenn Sie den `Terminate` Prozess unterbrechen, können Sie trotzdem die Beendigung von Instances erzwingen, indem Sie den [delete-auto-scaling-group](#) Befehl mit der Option `force delete` verwenden.
- Das Anhalten des `Terminate` Prozesses gilt nur für Instanzen, die sich derzeit im `InService` Status befinden. Es verhindert nicht die Beendigung von Instances in anderen Staaten, z. B., oder von Instances `Pending`, die nicht ordnungsgemäß aus dem Standby-Modus wieder aufgenommen werden können.
- Der `RemoveFromLoadBalancerLowPriority` Prozess kann ignoriert werden, wenn er in Aufrufen zur Beschreibung von Auto Scaling Scaling-Gruppen mit AWS CLI oder vorkommt SDKs. Dieser Prozess ist veraltet und wird nur aus Gründen der Abwärtskompatibilität beibehalten.

Prozess anhalten

Verwenden Sie eine der folgenden Methoden, um einen Prozess für eine Auto Scaling Scaling-Gruppe anzuhalten:

Console

Setzen Sie einen Prozess wie folgt aus:

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich `Auto Scaling Groups` aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte `Details` die Option `Erweiterte Konfigurationen, Bearbeiten`.
4. Wählen Sie für `Suspended Processes (Unterbrochene Prozesse)` den zu unterbrechenden Prozess aus.
5. Wählen Sie `Aktualisieren` aus.

AWS CLI

Verwenden Sie den folgenden [suspend-processes](#)-Befehl, um einzelne Prozesse anzuhalten.

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg --scaling-processes HealthCheck ReplaceUnhealthy
```

Um alle Prozesse anzuhalten, lassen Sie die `--scaling-processes`-Option wie folgt aus.

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg
```

Prozesse fortsetzen

Verwenden Sie eine der folgenden Methoden, um einen unterbrochenen Prozess für eine Auto Scaling Scaling-Gruppe wieder aufzunehmen:

Console

Setzen Sie einen ausgesetzten Prozess wie folgt fort:

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Erweiterte Konfigurationen, Bearbeiten.
4. Entfernen Sie für Suspended Processes (Unterbrochene Prozesse) den unterbrochenen Prozess.
5. Wählen Sie Aktualisieren aus.

AWS CLI

Um einen unterbrochenen Prozess wieder aufzunehmen, verwenden Sie den folgenden Befehl [resume-processes](#).

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg --scaling-processes HealthCheck
```

Um alle Prozesse fortzusetzen, lassen Sie die `--scaling-processes`-Option wie folgt aus.

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg
```

Wie sich unterbrochene Prozesse auf andere Prozesse auswirken

In den folgenden Abschnitten wird beschrieben, was passiert, wenn verschiedene Prozesse einzeln angehalten werden.

Themen

- [Launchist suspendiert](#)
- [Terminateist suspendiert](#)
- [AddToLoadBalancerist suspendiert](#)
- [AlarmNotificationist suspendiert](#)
- [AZRebalanceist suspendiert](#)
- [HealthCheckist suspendiert](#)
- [InstanceRefreshist suspendiert](#)
- [ReplaceUnhealthyist suspendiert](#)
- [ScheduledActionsist suspendiert](#)
- [Weitere Überlegungen](#)

Launchist suspendiert

- `AlarmNotification` ist immer noch aktiv, aber Ihre Auto-Scaling-Gruppe kann keine Aufskalierungs-Aktivitäten für Alarme initiieren, die verletzt sind.
- `ScheduledActions` ist aktiv, aber Ihre Auto-Scaling-Gruppe kann keine Aufskalierungs-Aktivitäten für geplante Aktionen initiieren, die auftreten.
- `AZRebalance` hört auf, die Gruppe auszugleichen.
- `ReplaceUnhealthy` beendet weiterhin fehlerhafte Instances, startet jedoch keine Ersetzungen. Wenn Sie den Prozess Launch wieder aufnehmen, ersetzt Amazon EC2 Auto Scaling sofort alle Instances, die es während der Zeit, in der Launch ausgesetzt war, beendet hat.
- `InstanceRefresh` ersetzt keine Instances.

Terminate ist suspendiert

- `AlarmNotification` ist immer noch aktiv, aber Ihre Auto Scaling-Gruppe kann keine Skalierungsaktivitäten für Alarme einleiten, bei denen ein Verstoß vorliegt.
- `ScheduledActions` ist aktiv, aber Ihre Auto Scaling-Gruppe kann keine Skalierungsaktivitäten für geplante Aktionen einleiten.
- `AZRebalance` ist noch aktiv, funktioniert jedoch nicht ordnungsgemäß. Es können neue Instances gestartet werden, ohne dass alte beendet werden. Dies kann dazu führen, dass Ihre Auto-Scaling-Gruppe auf eine Größe anwächst, die ihre Höchstgröße um bis zu 10 Prozent übersteigt, da dies während Aktivitäten zur Wiederherstellung des Gleichgewichts vorübergehend zulässig ist. Ihre Auto-Scaling-Gruppe könnte diese die Höchstgröße überschreitende Größe beibehalten, bis Sie den `Terminate`-Prozess fortsetzen.
- `ReplaceUnhealthy` ist inaktiv, jedoch nicht `HealthCheck`. Wenn `Terminate` fortgesetzt wird, wird der Prozess `ReplaceUnhealthy` sofort ausgeführt. Wenn Instances als fehlerhaft markiert wurden, während `Terminate` unterbrochen war, werden sie sofort ersetzt.
- `InstanceRefresh` ersetzt keine Instances.

AddToLoadBalancer ist suspendiert

- Amazon EC2 Auto Scaling startet die Instances, fügt sie aber nicht zur Load Balancer-Zielgruppe oder Classic Load Balancer hinzu. Wird der Prozess `AddToLoadBalancer` fortgesetzt, fügt es Instances beim Start wieder zum Load Balancer hinzu. Allerdings fügt es keine Instances hinzu, die gestartet wurden, als der Prozess ausgesetzt war. Diese Instances müssen Sie manuell anmelden.

AlarmNotification ist suspendiert

- Amazon EC2 Auto Scaling ruft keine Skalierungsrichtlinien auf, wenn ein CloudWatch Alarmschwellenwert überschritten wird. Wenn Sie `AlarmNotification` fortsetzen, berücksichtigt Amazon EC2 Auto Scaling Richtlinien mit Alarmschwellenwerten aus, die derzeit überschritten werden.

AZRebalance ist suspendiert

- Amazon EC2 Auto Scaling versucht nicht, Instances nach bestimmten Ereignissen neu zu verteilen. Wenn jedoch ein Scale-Out- oder Scale-in-Ereignis eintritt, versucht der

Skalierungsprozess trotzdem, die Availability Zones auszugleichen. Beim Scale-Out werden beispielsweise Instances in der Availability Zone mit den wenigsten Instances gestartet. Wenn die Gruppe aus dem Gleichgewicht gerät, während AZRebalance unterbrochen ist, und Sie dies fortsetzen, versucht Amazon EC2 Auto Scaling, die Gruppe wieder auszugleichen. Zuerst wird Launch und anschließend Terminate aufgerufen.

- Warme Pools sind nicht betroffen, wenn die Einstellung unterbrochen AZRebalance ist.

HealthCheckist suspendiert

- Amazon EC2 Auto Scaling beendet das Markieren von Instances als fehlerhaft infolge der Zustandsprüfungen durch EC2 und Elastic Load Balancing. Ihre benutzerdefinierten Zustandsprüfungen funktionieren weiterhin ordnungsgemäß. Wenn Sie HealthCheck unterbrochen haben, können Sie den Zustand von Instances in Ihrer Gruppe bei Bedarf manuell festlegen und sie durch ReplaceUnhealthy ersetzen lassen.

InstanceRefreshist suspendiert

- Amazon EC2 Auto Scaling beendet das Ersetzen von Instances als Ergebnis einer Instance-Aktualisierung. Wenn eine Instance-Aktualisierung ausgeführt wird, unterbricht dies den Vorgang, ohne ihn abubrechen.

ReplaceUnhealthyist suspendiert

- Amazon EC2 Auto Scaling stoppt das Ersetzen von Instances, die als fehlerhaft markiert wurden. Instances, die Zustandsprüfungen von EC2 oder Elastic Load Balancing nicht bestehen, werden weiterhin als fehlerhaft markiert. Sobald Sie den Prozess ReplaceUnhealthy fortsetzen, ersetzt Amazon EC2 Auto Scaling die Instances, die als fehlerhaft markiert wurden, während dieser Prozess unterbrochen war. Der ReplaceUnhealthy-Prozess ruft zuerst Terminate und dann Launch auf.

ScheduledActionsist suspendiert

- Amazon EC2 Auto Scaling führt keine geplanten Aktionen aus, die während des Aussetzungszeitraums geplant sind. Wenn Sie ScheduledActions wieder aufnehmen,

berücksichtigt Amazon EC2 Auto Scaling nur geplante Aktionen, deren geplante Zeit noch nicht abgelaufen ist.

Weitere Überlegungen

Darüber hinaus, wenn `Launch` oder `Terminate` ausgesetzt sind, funktionieren die folgenden Funktionen möglicherweise nicht richtig:

- **Maximale Instanzlebensdauer** — Wenn `Launch` oder gesperrt `Terminate` sind, kann die Funktion zur maximalen Instanzlebensdauer keine Instanzen ersetzen.
- **Spot-Instance-Unterbrechungen** — Wenn `Terminate` die Spot-Instances ausgesetzt sind und Ihre Auto Scaling Scaling-Gruppe über Spot-Instances verfügt, können diese trotzdem beendet werden, falls Spot-Kapazitäten nicht mehr verfügbar sind. Während `Launch` ausgesetzt ist, kann Amazon EC2 Auto Scaling keine Ersatz-Instances aus einem anderen Spot-Instance-Pool oder aus demselben Spot-Instance-Pool starten, wenn dieser wieder verfügbar ist.
- **Kapazitätsausgleich** — Wenn die Einstellung unterbrochen `Terminate` ist und Sie `Capacity Rebalancing` verwenden, um Spot-Instance-Unterbrechungen zu beheben, kann der Amazon EC2 Spot-Service Instances trotzdem beenden, falls Spot-Kapazität nicht mehr verfügbar ist. Wenn `Launch` ausgesetzt ist, kann Amazon EC2 Auto Scaling keine Ersatz-Instances aus einem anderen Spot-Instance-Pool oder aus demselben Spot-Instance-Pool starten, wenn dieser wieder verfügbar ist.
- **Instances anhängen und trennen** — Wenn `Launch` und suspendiert `Terminate` sind, können Sie Instances trennen, die an Ihre Auto Scaling Scaling-Gruppe angehängt sind, aber solange gesperrt `Launch` ist, können Sie keine neuen Instances an die Gruppe anhängen.
- **Standby-Instances** — Wenn `Launch` und suspendiert `Terminate` sind, können Sie eine Instance in den Standby Status versetzen, aber solange sie suspendiert `Launch` ist, können Sie eine Instance im Standby Status nicht wieder in Betrieb nehmen.

Überwachen Sie Ihre Amazon EC2 Auto Scaling Scaling-Gruppen

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon EC2 Auto Scaling und Ihrer AWS Cloud Lösungen. AWS bietet die folgenden Überwachungstools, um Amazon EC2 Auto Scaling zu beobachten, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen:

Health checks (Zustandsprüfungen)

Amazon EC2 Auto Scaling führt regelmäßig Zustandsprüfungen für die Instances in Ihrer Auto-Scaling-Gruppe durch. Wenn eine Instance diese Zustandsprüfungen nicht besteht, wird sie als fehlerhaft markiert und beendet, während Amazon EC2 Auto Scaling eine neue Instance als Ersatz startet. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

AWS Health Dashboard

Das Health Dashboard zeigt Informationen an und bietet auch Benachrichtigungen, die bei Änderungen im Zustand von AWS Ressourcen ausgelöst werden. Diese Informationen werden auf zweierlei Weise dargestellt: in einem Dashboard, das kürzliche und kommende Ereignisse nach Kategorie sortiert anzeigt, und in einem vollständigen Ereignisprotokoll, das alle Ereignisse der letzten 90 Tage enthält. Weitere Informationen finden Sie unter [Health Dashboard Benachrichtigungen für Amazon EC2 Auto Scaling](#).

CloudTrail

Mit AWS CloudTrail können Sie die Aufrufe der Amazon EC2 Auto Scaling Scaling-API von oder in Ihrem AWS-Konto Namen verfolgen. CloudTrail speichert die Informationen in Protokolldateien im Amazon S3 S3-Bucket, den Sie angeben. Mit diesen Protokolldateien können Sie die Aktivitäten Ihrer Auto-Scaling-Gruppen überwachen. Protokolle enthalten Informationen dazu, welche Anforderungen erfolgt sind, zu den Quell-IP-Adressen, von denen die Anforderungen kamen, wer die Anforderung gestellt hat, wann die Anforderung erfolgt ist usw. Weitere Informationen finden Sie unter [Protokollieren Sie Amazon EC2 Auto Scaling API-Aufrufe mit AWS CloudTrail](#).

Protokollerfassung für Ihre Amazon EC2 Instances

Sie können CloudWatch es verwenden, um Protokolle von den Betriebssystemen für Ihre EC2-Instances zu sammeln. Weitere Informationen finden Sie unter [Erfassung von Metriken und Protokollen von Amazon EC2 EC2-Instances und lokalen Servern mit dem CloudWatch Agenten](#) und [An CloudWatch Logs gesendete Protokolldaten anzeigen](#) im CloudWatch Amazon-Benutzerhandbuch.

Informationen zu anderen AWS Diensten, die Ihnen bei der Protokollierung und Erfassung von Daten über Ihre Workloads helfen können, finden Sie im [Leitfaden zur Protokollierung und Überwachung für Anwendungsbesitzer](#) in der AWS Prescriptive Guidance.

Amazon CloudWatch

Amazon CloudWatch unterstützt Sie bei der Analyse von Protokollen und bei der Überwachung der Kennzahlen Ihrer AWS Ressourcen und gehosteten Anwendungen in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarme festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Beispielsweise können Sie benachrichtigt werden, wenn die Netzwerkaktivität plötzlich höher oder niedriger als der erwartete Wert einer Metrik ist. Weitere Informationen über die Verwendung dieses Services zur Überwachung der Metriken Ihrer Auto-Scaling-Gruppen und -Instances finden Sie unter [Überwachen Sie CloudWatch Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances](#).

CloudWatch verfolgt auch AWS API-Nutzungsmetriken für Amazon EC2 Auto Scaling. Sie können diese Metriken verwenden, um Alarme zu konfigurieren, die Sie benachrichtigen, wenn Ihr API-Aufrufvolumen einen von Ihnen definierten Schwellenwert überschreitet. Weitere Informationen finden Sie unter [AWS Nutzungsmetriken](#) im CloudWatch Amazon-Benutzerhandbuch.

AWS Compute Optimizer

Compute Optimizer bietet Empfehlungen für Amazon-EC2-Instances, die Ihnen bei der Entscheidung helfen können, zu einem neuen Instance-Typ zu wechseln. Er analysiert, ob der Instance-Typ einer Auto-Scaling-Gruppe optimal ist und generiert Empfehlungen, um die Kosten zu senken und die Leistung Ihrer Workloads zu verbessern. Weitere Informationen finden Sie unter [Holen Sie sich Empfehlungen zum Instanztyp mit AWS Compute Optimizer](#).

Amazon EventBridge

Amazon EventBridge ist ein serverloser Event-Bus-Service, der es einfach macht, Ihre Anwendungen mit Daten aus einer Vielzahl von Quellen zu verbinden. EventBridge liefert einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, Software-as-a-Service (SaaS-) Anwendungen und AWS Diensten und leitet diese Daten an Ziele wie Lambda weiter. Auf diese Weise können Sie Ereignisse überwachen, die in Services auftreten, und ereignisgesteuerte Architekturen erstellen. Weitere Informationen finden Sie unter [Wird EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen verwendet.](#)

AWS Security Hub CSPM

Verwenden Sie [AWS Security Hub CSPM](#), um Ihre Nutzung von Amazon EC2 Auto Scaling im Hinblick auf bewährte Sicherheitsmethoden zu überwachen. Security Hub CSPM verwendet detektive Sicherheitskontrollen, um Ressourcenkonfigurationen und Sicherheitsstandards zu bewerten und Sie bei der Einhaltung verschiedener Compliance-Frameworks zu unterstützen. Weitere Informationen zur Verwendung von Security Hub CSPM zur Evaluierung von Amazon EC2 Auto Scaling-Ressourcen finden Sie unter [Amazon EC2 Auto Scaling-Steuerelemente im Benutzerhandbuch](#).AWS Security Hub

Amazon Simple Notification Service

Sie können Auto-Scaling-Gruppen so konfigurieren, dass sie Amazon SNS Benachrichtigungen senden, wenn Amazon-EC2-Auto-Scaling startet. Weitere Informationen finden Sie unter [Amazon SNS SNS-Benachrichtigungsoptionen für Amazon EC2 Auto Scaling](#).

Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe

Amazon EC2 Auto Scaling überwacht kontinuierlich den Integritätsstatus von Instances in einer Auto Scaling Scaling-Gruppe, um die gewünschte Kapazität aufrechtzuerhalten.

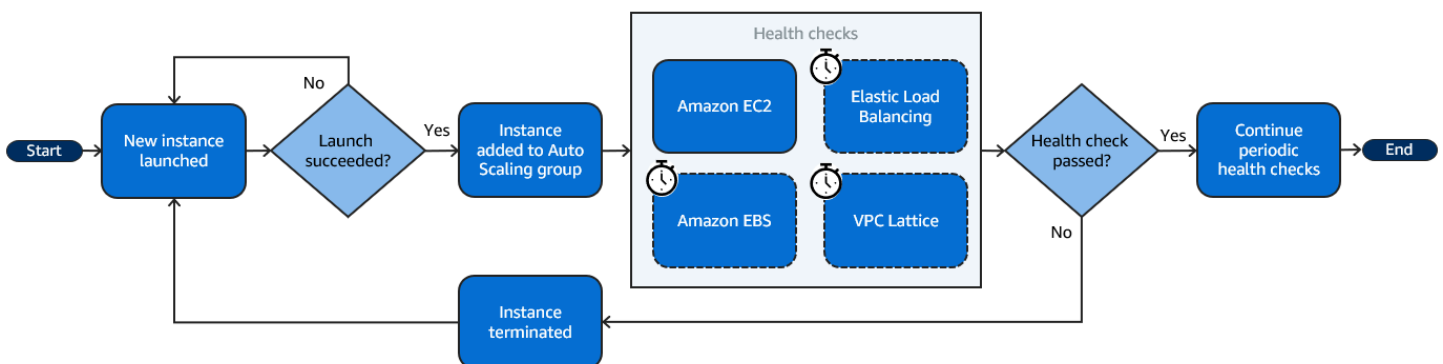
Alle Instances in einer Auto Scaling Scaling-Gruppe beginnen mit einem Healthy Status. Instances werden als fehlerfrei betrachtet, bis Amazon EC2 Auto Scaling benachrichtigt wird, dass sie fehlerhaft sind. Sie kann Benachrichtigungen von verschiedenen Quellen empfangen, wenn eine Instanz defekt ist und ersetzt werden muss. Diese Quellen umfassen u. a. folgende:

- Amazon EC2
- Elastic Load Balancing
- VPC-Gitter

- Amazon EBS
- Benutzerdefinierte Integritätsprüfungen, die Sie definieren

Wenn Amazon EC2 Auto Scaling feststellt, dass eine InService Instance fehlerhaft ist, wird sie durch eine neue Instance ersetzt, um die gewünschte Kapazität der Gruppe aufrechtzuerhalten. Die neue Instance wird mit den aktuellen Einstellungen der Auto-Scaling-Gruppe und der zugehörigen Startvorlage oder Startkonfiguration gestartet.

Das folgende Flussdiagramm veranschaulicht den Prozess des Starts einer neuen Instance in einer Auto Scaling Scaling-Gruppe. Es beginnt mit dem Starten der Instance. Wenn der Start erfolgreich ist, wird die Instance der Auto Scaling Scaling-Gruppe hinzugefügt. Anschließend führt Amazon EC2 Auto Scaling mithilfe der integrierten Amazon EC2 EC2-Statuschecks Integritätsprüfungen für die Instance und nach einer Übergangszeit alle optionalen Zustandsprüfungen durch, die Sie für die Gruppe aktiviert haben. Diese Zustandsprüfungen werden regelmäßig fortgesetzt. Wenn eine der Integritätsprüfungen fehlschlägt, wird die Instanz ersetzt.



Fehlerhafte Instances können auch auftreten, wenn eine Instance unerwartet beendet wird, z. B. aufgrund einer Unterbrechung der Spot-Instance oder einer manuellen Kündigung durch einen Benutzer. Auch in diesen Fällen startet Amazon EC2 Auto Scaling automatisch eine Ersatz-Instance, um die gewünschte Kapazität aufrechtzuerhalten.

Inhalt

- [Über Zustandsprüfungen Ihrer Auto-Scaling-Gruppe](#)
- [Legen Sie die Wartefrist für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#)
- [Überwachen Sie Auto Scaling Scaling-Instances mit beeinträchtigten Amazon EBS-Volumes mithilfe von Zustandsprüfungen](#)
- [Richten Sie eine benutzerdefinierte Integritätsprüfung für Ihre Auto Scaling Scaling-Gruppe ein](#)
- [Anzeigen des Grundes für Fehler bei Zustandsprüfung](#)

- [Fehlerbehebung bei fehlerhaften Instances in Amazon EC2 Auto Scaling](#)

Über Zustandsprüfungen Ihrer Auto-Scaling-Gruppe

Dieses Thema bietet einen Überblick über die verfügbaren Zustandsprüfungstypen und beschreibt die wichtigsten Überlegungen zur Integration von Amazon EC2 Auto Scaling Scaling-Zustandsprüfungen in Ihre Anwendungen.

Inhalt

- [Health check type \(Typ der Zustandsprüfung\)](#)
- [Zustandsprüfungen von Amazon EC2](#)
- [Zustandsprüfungen von Elastic Load Balancing](#)
- [VPC-Lattice-Zustandsprüfungen](#)
- [Wie Amazon EC2 Auto Scaling Ausfallzeiten minimiert](#)
- [Gesundheitschecks für Instanzen in einem warmen Pool](#)
- [Überlegungen zur Zustandsprüfung](#)

Health check type (Typ der Zustandsprüfung)

Amazon EC2 Auto Scaling kann den Integritätsstatus einer InService Instance mithilfe einer oder mehrerer der folgenden Zustandsprüfungen ermitteln:

Typ der Zustandsprüfung	Was es überprüft
Amazon-EC2-Zustandsprüfungen und geplante Ereignisse	<ul style="list-style-type: none"> • Überprüft, ob die Instance läuft. • Prüft auf zugrunde liegende Hardware- oder Softwareprobleme, die die Instanz beeinträchtigen könnten. <p>Dies ist der standardmäßige Zustandsprüfungstyp für eine Auto-Scaling-Gruppe.</p>
Zustandsprüfungen von Elastic Load Balancing	<ul style="list-style-type: none"> • Überprüft, ob der Load Balancer die Instance als fehlerfrei meldet, und bestätigt, ob die Instance für die Bearbeitung von Anfragen verfügbar ist.

Typ der Zustandsprüfung	Was es überprüft
	Um diesen Integritätsprüfungstyp auszuführen, müssen Sie ihn für Ihre Auto Scaling Scaling-Gruppe aktivieren.
VPC-Lattice-Zustandsprüfungen	<ul style="list-style-type: none"> Überprüft, ob VPC Lattice die Instance als fehlerfrei meldet, und bestätigt, ob die Instance für die Bearbeitung von Anfragen verfügbar ist. <p>Um diesen Integritätsprüfungstyp auszuführen, müssen Sie ihn für Ihre Auto Scaling Scaling-Gruppe aktivieren.</p>
Amazon-EBS-Zustandsprüfungen	<ul style="list-style-type: none"> Überprüft, ob EBS-Volumes erreichbar sind, und besteht die I/O Statusprüfungen. <p>Um diesen Integritätsprüfungstyp auszuführen, müssen Sie ihn für Ihre Auto Scaling Scaling-Gruppe aktivieren.</p>
Benutzerdefinierte Zustandsprüfungen	<ul style="list-style-type: none"> Sucht gemäß Ihren benutzerdefinierten Integritätsprüfungen nach anderen Problemen, die auf Integritätsprobleme der Instance hinweisen könnten.

Zustandsprüfungen von Amazon EC2

Nachdem eine Instance gestartet wurde, wird sie an die Auto-Scaling-Gruppe angefügt und wechselt in den Zustand `InService`. Weitere Informationen über die verschiedenen Lebenszyklus-Statusse der Instances in einer Auto-Scaling-Gruppe finden Sie unter [Instance-Lebenszyklus bei Amazon EC2 Auto Scaling](#).

Amazon EC2 Auto Scaling überprüft durch eine regelmäßige Zustandsprüfung aller Instances innerhalb der Auto-Scaling-Gruppe, ob sie ausgeführt werden und sich in einem guten Zustand befinden.

Statusüberprüfungen

Amazon EC2 Auto Scaling verwendet standardmäßig die Ergebnisse der Amazon EC2-Instance-Statusprüfungen und System-Statusprüfungen, um den Zustand einer Instance zu bestimmen. Wenn sich die Instance in einem anderen Amazon-EC2-Status als `running` befindet oder der Systemstatus

impaired ist, betrachtet Amazon EC2 Auto Scaling die Instance als fehlerhaft und ersetzt sie. Dies gilt auch, wenn die Instance einen der folgenden Zustände aufweist:

- stopping
- stopped
- shutting-down
- terminated

Die Amazon EC2-Statusprüfungen erfordern keine spezielle Konfiguration und sind stets aktiviert. Weitere Informationen finden Sie unter [Arten von Statusprüfungen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Important

Amazon EC2 Auto Scaling lässt die Statusprüfungen gelegentlich ausfallen, ohne Maßnahmen zu ergreifen. Wenn eine Statusüberprüfung fehlschlägt, wartet Amazon EC2 Auto Scaling einige Minuten, AWS bis das Problem behoben ist. Es markiert eine Instance nicht sofort als Unhealthy, wenn ihr Status für die Zustandsprüfungen impaired wird. Darüber hinaus markiert EC2 Auto Scaling die Instance nicht so, als Unhealthy ob eine Statusüberprüfung zurückgegeben insufficient-data würde.

Wenn Amazon EC2 Auto Scaling jedoch erkennt, dass sich eine Instance nicht mehr im Status running befindet, wird diese Situation umgehend als Fehler behandelt. In diesem Fall markiert es die Instance sofort als Unhealthy und ersetzt sie.

Geplante Ereignisse

Amazon EC2 kann gelegentlich Ereignisse auf Ihren Instances planen, die nach einem bestimmten Zeitstempel ausgeführt werden. Weitere Informationen finden Sie unter [Geplante Ereignisse für Ihre Instances](#) im Benutzerhandbuch für Amazon EC2.

Wenn eine Ihrer Instances von einem geplanten Ereignis betroffen ist, betrachtet Amazon EC2 Auto Scaling die Instance als fehlerhaft und ersetzt sie. Die Instanz wird erst heruntergefahren, wenn das im Zeitstempel angegebene Datum und die Uhrzeit erreicht sind.

Zustandsprüfungen von Elastic Load Balancing

Wenn Sie Elastic Load Balancing Health Checks für Ihre Auto Scaling-Gruppe aktivieren, kann Amazon EC2 Auto Scaling die Ergebnisse dieser Zustandsprüfungen verwenden, um den Integritätsstatus einer Instance zu ermitteln.

Bevor Sie Elastic Load Balancing Health Checks für Ihre Auto Scaling-Gruppe aktivieren können, müssen Sie einen Elastic Load Balancing Load Balancer konfigurieren und dafür eine Integritätsprüfung konfigurieren, um festzustellen, ob Ihre Instances fehlerfrei sind. Weitere Informationen finden Sie unter [Bereiten Sie das Anhängen eines Elastic Load Balancing Load Balancers vor](#).

Nachdem Sie den Load Balancer Ihrer Auto Scaling-Gruppe hinzugefügt haben, passiert Folgendes:

- Amazon EC2 Auto Scaling registriert die Instances in der Auto-Scaling-Gruppe beim Load Balancer.
- Nachdem eine Instance die Registrierung beendet hat, wechselt sie in den Status `InService` und wird für die Verwendung mit dem Load Balancer verfügbar.

Standardmäßig ignoriert Amazon EC2 Auto Scaling die Ergebnisse der Zustandsprüfungen des Elastic Load Balancing. Wenn Sie diese Zustandsprüfungen für Ihre Auto Scaling-Gruppe aktiviert haben und Elastic Load Balancing eine registrierte Instance als `Unhealthy` meldet, markiert Amazon EC2 Auto Scaling die Instance `Unhealthy` bei der nächsten regelmäßigen Integritätsprüfung und ersetzt sie.

Wenn der Connection Draining (Verzögerte Deregistrierung) für Ihren Load Balancer aktiviert ist, wartet Amazon EC2 Auto Scaling, bis die laufenden Anforderungen abgeschlossen werden oder das maximale Zeitlimit abgelaufen ist, bevor die fehlerhaften Instances beendet werden.

Note

Anweisungen, wie Sie den Load Balancer anhängen und Elastic Load Balancing Health Checks für Ihre Auto Scaling-Gruppe aktivieren, finden Sie unter [Fügen Sie Ihrer Auto Scaling-Gruppe einen Elastic Load Balancing Load Balancer hinzu](#).

Wenn Sie Elastic Load Balancing Health Checks für eine Gruppe aktivieren, kann Amazon EC2 Auto Scaling Instances ersetzen, die Elastic Load Balancing als fehlerhaft meldet, aber

erst, nachdem sich der Load Balancer im Status befindet. InService Weitere Informationen finden Sie unter [Überprüfen des Anhangsstatus Ihres Load Balancers](#).

VPC-Lattice-Zustandsprüfungen

Standardmäßig ignoriert Amazon EC2 Auto Scaling die Ergebnisse der VPC-Lattice-Zustandsprüfungen. Sie können diese Integritätsprüfungen optional für Ihre Auto Scaling Scaling-Gruppe aktivieren. Wenn Sie dies getan haben und VPC-Lattice eine registrierte Instance als `Unhealthy` meldet, markiert Amazon EC2 Auto Scaling die Instance bei der nächsten regelmäßigen Zustandsprüfung als `Unhealthy` und ersetzt sie. Der Prozess des Registrierens von Instances und der anschließenden Überprüfung ihres Zustands entspricht der Funktionsweise von Elastic-Load-Balancing-Zustandsprüfungen.

Note

Anweisungen zum Anhängen der VPC Lattice-Zielgruppe und zum Aktivieren der VPC Lattice-Zustandsprüfungen für Ihre Auto Scaling Scaling-Gruppe finden Sie unter [Hinzufügen einer VPC-Lattice-Zielgruppe zu Ihrer Auto-Scaling-Gruppe](#)

Wenn Sie VPC Lattice-Integritätsprüfungen für eine Gruppe aktivieren, kann Amazon EC2 Auto Scaling Instances ersetzen, die VPC Lattice als fehlerhaft meldet, aber erst, nachdem sich die Zielgruppe im Status befindet. InService Weitere Informationen finden Sie unter [Überprüfen Sie den Anhangsstatus Ihrer VPC Lattice-Zielgruppe](#).

Wie Amazon EC2 Auto Scaling Ausfallzeiten minimiert

Standardmäßig werden neue Instances gleichzeitig bereitgestellt, wenn Ihre vorhandenen Instances beendet werden. Dadurch können neue Anfragen möglicherweise nicht akzeptiert werden, bis die neuen Instances voll funktionsfähig sind.

Wenn Amazon EC2 Auto Scaling feststellt, dass Instances nicht mehr laufen (oder sie `Unhealthy` mit dem [set-instance-health](#) Befehl markiert wurden), werden sie sofort ersetzt. Wenn jedoch andere Instances als fehlerhaft eingestuft werden, verwendet Amazon EC2 Auto Scaling den folgenden Ansatz, um sich von Ausfällen zu erholen. Dieser Ansatz minimiert Ausfallzeiten, die aufgrund vorübergehender Probleme oder falsch konfigurierter Integritätsprüfungen auftreten können.

- Wenn gerade eine Skalierungsaktivität läuft und Ihre Auto Scaling-Gruppe die gewünschte Kapazität um 10 Prozent oder mehr unterschreitet, wartet Amazon EC2 Auto Scaling auf die laufende Skalierungsaktivität, bevor die fehlerhaften Instances ersetzt werden.
- Bei der Skalierung wartet Amazon EC2 Auto Scaling darauf, dass die Instances eine erste Zustandsprüfung bestehen. Es wartet auch darauf, dass die Standard-Instance-Aufwärmphase beendet ist, um sicherzustellen, dass die neuen Instances bereit sind.
- Nachdem die Instances das Warmlaufen abgeschlossen haben und die Gruppe auf mehr als 90 Prozent der gewünschten Kapazität angewachsen ist, ersetzt Amazon EC2 Auto Scaling die fehlerhaften Instances wie folgt:
 - Amazon EC2 Auto Scaling ersetzt jeweils nur bis zu 10 Prozent der gewünschten Kapazität der Gruppe. Dies geschieht, bis alle fehlerhaften Instances ersetzt wurden.
 - Beim Ersetzen von Instances wird gewartet, bis die neuen Instances eine erste Zustandsprüfung bestanden haben. Es wartet auch darauf, dass das Aufwärmen der Standard-Instance abgeschlossen ist, bevor es fortfährt.

Note

- Wenn die Größe einer Auto Scaling-Gruppe so klein ist, dass der resultierende Wert von 10 Prozent kleiner als eins ist, ersetzt Amazon EC2 Auto Scaling stattdessen die fehlerhaften Instances nacheinander. Dies kann zu Ausfallzeiten für die Gruppe führen.
- Sie können den Standardwert von 10 Prozent ändern, indem Sie [eine Instance-Wartungsrichtlinie festlegen](#), um die Geschwindigkeit zu ändern, mit der Auto Scaling fehlerhafte Instances ersetzt. Auto Scaling kann jedoch immer noch die Geschwindigkeit drosseln, mit der Instances als fehlerhaft markiert werden.

Wenn beispielsweise alle Instances in einer Auto Scaling-Gruppe von Elastic Load Balancing als fehlerhaft gemeldet werden und sich der Load Balancer im InService Status befindet, markiert Amazon EC2 Auto Scaling möglicherweise weniger Instances gleichzeitig als fehlerhaft. Dies kann dazu führen, dass viel weniger Instances gleichzeitig ersetzt werden als die 10 Prozent, die in anderen Szenarien angewendet wurden. Dadurch haben Sie Zeit, das Problem zu beheben, ohne dass Amazon EC2 Auto Scaling automatisch die gesamte Gruppe beendet.

Gesundheitschecks für Instanzen in einem warmen Pool

Amazon EC2 Auto Scaling führt auch Integritätsprüfungen für Instances in einem warmen Pool durch. Weitere Informationen finden Sie unter [Anzeigen des Status der Zustandsprüfung und dem Grund für Zustandsprüfungsfehler](#).

Überlegungen zur Zustandsprüfung

Im Folgenden finden Sie Überlegungen zur Verwendung von Amazon EC2 Auto Scaling Scaling-Zustandsprüfungen.

- Sie Lebenszyklus-Hooks verwenden, wenn auf der Instance, die beendet wird, oder auf der Instance, die gestartet wird, etwas passieren muss. Mithilfe dieser Hooks können Sie eine benutzerdefinierte Aktion ausführen, wenn Amazon EC2 Auto Scaling Instances startet oder beendet. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).
- Amazon EC2 Auto Scaling bietet keine Möglichkeit, die Amazon EC2-Statusprüfungen und geplante Ereignisse aus den Zustandsprüfungen auszuschließen. Wenn Sie nicht möchten, dass Instances ersetzt werden, empfehlen wir Ihnen, den `ReplaceUnhealthy`- und `HealthCheck`-Prozess für einzelne Auto-Scaling-Gruppen auszusetzen. Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).
- Um den Integritätsstatus einer fehlerhaften Instance manuell wieder auf den Status `Healthy` zurückzusetzen, können Sie versuchen, den `set-instance-health` Befehl zu verwenden. Wenn Sie eine Fehlermeldung erhalten, liegt das wahrscheinlich daran, dass die Instance bereits beendet ist. Im Allgemeinen ist das Zurücksetzen des Integritätsstatus einer Instanz `Healthy` mit dem `set-instance-health` Befehl nur dann sinnvoll, wenn entweder der `ReplaceUnhealthy` Prozess oder der `Terminate` Prozess unterbrochen ist.
- Wenn Sie Fehler bei einer Instanz beheben müssen, ohne dass es zu Störungen durch Integritätsprüfungen kommt, können Sie die Instanz in den `Standby` Status versetzen. Amazon EC2 Auto Scaling führt keine Integritätsprüfungen für Instances durch, die sich im `Standby` Status befinden, bis Sie die Instances wieder in Betrieb nehmen. Weitere Informationen finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#).
- Wenn die Instance beendet wird, werden alle zugehörigen Elastic IP-Adressen von ihr getrennt und der neuen Instance nicht automatisch zugeordnet. Sie müssen die elastische IP-Adressen manuell mit der neuen Instance verknüpfen oder dies automatisch mit einer auf Lebenszyklus-Hook-basierter Lösung tun. Weitere Informationen finden Sie unter [Elastische IP-Adressen](#) im Benutzerhandbuch für Amazon EC2.

- In ähnlicher Weise werden beim Beenden der Instance ihre zugehörigen EBS-Volumes getrennt (oder je nach `DeleteOnTermination`-Attribut des Volumes gelöscht). Sie müssen diese EBS-Volumes manuell an die neue Instance anhängen oder dies automatisch mit einer Lebenszyklus-Hook-basierten Lösung tun. Weitere Informationen finden Sie unter [Anfügen eines Amazon-EBS-Volumes an eine Instance](#) im Amazon-EBS-Benutzerhandbuch.

Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest

Wenn durch eine Amazon EC2 Auto Scaling Zustandsprüfung festgestellt wird, dass eine `InService`-Instance fehlerhaft ist, wird die Instance durch eine neue Instance ersetzt. Die Frist für die Zustandsprüfung gibt die Mindestdauer (in Sekunden) an, die eine neue Instance in Betrieb bleiben muss, bevor sie beendet wird, wenn sie als fehlerhaft erkannt wird.

In einem bestimmten Anwendungsfall kann es erforderlich sein, auf Amazon EC2 Auto Scaling zu warten, wenn die Zustandsprüfungen für Elastic Load Balancing fehlschlagen und die Instance noch initialisiert wird. Die Zustandsprüfungen von Elastic Load Balancing werden parallel ausgeführt, beginnend mit der Registrierung der Instance beim Load Balancer. Die Übergangsfrist verhindert, dass Amazon EC2 Auto Scaling Ihre neu gestarteten Instances markiert `Unhealthy` und unnötig beendet, wenn sie diese Zustandsprüfungen nicht sofort bestehen, nachdem sie den Status erreicht haben. `InService`

In der Konsole beträgt der Kulanzzzeitraum für die Zustandsprüfung standardmäßig 300 Sekunden, wenn Sie eine Auto-Scaling-Gruppe erstellen. Der Standardwert ist 0 Sekunden, wenn Sie eine Auto Scaling Scaling-Gruppe mit dem AWS CLI oder einem SDK erstellen. Ein Wert von 0 deaktiviert die Nachfrist für Zustandsprüfungen.

Wenn Sie diesen Wert zu hoch festlegen, wird die Effektivität der Amazon EC2 Auto Scaling-Zustandsprüfungen verringert. Wenn Sie einen Lebenszyklus-Hook für den Instance-Start verwenden, können Sie den Wert der Übergangsfrist für die Zustandsprüfung auf 0 festlegen. Mit Lifecycle-Hooks bietet Amazon EC2 Auto Scaling eine Möglichkeit, sicherzustellen, dass Instances immer initialisiert werden, bevor sie in den Status `InService` gelangen. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).

Die Nachfrist gilt für die folgenden Fälle:

- Neu gestartete Instances
- Instances, die nach dem Bereitschaftsmodus wieder in Betrieb genommen werden

- Instances, die Sie manuell an die Gruppe anhängen

Important

Wenn Amazon EC2 Auto Scaling erkennt, dass sich eine Instance während der Nachfrist für Zustandsprüfungen nicht mehr im Zustand `running` befindet, markiert Amazon EC2 Auto Scaling sie als `Unhealthy` und ersetzt sie. Wenn Sie beispielsweise eine Instance in einer Auto-Scaling-Gruppe beenden, wird sie als `Unhealthy` markiert und ersetzt.

Legen Sie die Wartezeit für die Zustandsprüfung einer Gruppe fest

Sie können die Wartezeit für die Zustandsprüfung für neue und vorhandene Auto-Scaling-Gruppen festlegen.

Console

Um den Kulanzzzeitraum für die Integritätsprüfung für eine neue Gruppe zu ändern

Wenn Sie die Auto Scaling Scaling-Gruppe erstellen, geben Sie die Zeitspanne (in Sekunden) auf der Seite Erweiterte Optionen konfigurieren unter Integritätsprüfungen und Kulanzzzeitraum Health Integritätsprüfungen ein. So lange muss Amazon EC2 Auto Scaling warten, bevor es den Integritätsstatus einer Instance überprüft, nachdem sie den `InService` Status erreicht hat.

AWS CLI

Um den Kulanzzzeitraum für die Integritätsprüfung für eine neue Gruppe zu ändern

Fügen Sie die `--health-check-grace-period` Option dem [create-auto-scaling-group](#) Befehl hinzu. Im folgenden Beispiel wird der Karenzzeit für die Zustandsprüfung mit einem Wert von **60** Sekunden für eine neue Auto-Scaling-Gruppe mit dem Namen **my-asg** konfiguriert.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
--health-check-grace-period 60 ...
```

Console

Um den Kulanzzzeitraum für die Integritätsprüfung für eine bestehende Gruppe zu ändern

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die AWS-Region aus, in der Sie Ihre Auto-Scaling-Gruppe erstellt haben.
3. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Wählen Sie auf der Registerkarte Details die Option Zustandsprüfungen, Bearbeiten aus.
5. Geben Sie unter Karenzzeit für die Zustandsprüfung die Zeit in Sekunden ein. So lange muss Amazon EC2 Auto Scaling warten, bevor es den Integritätsstatus einer Instance überprüft, nachdem sie den InService Status erreicht hat.
6. Wählen Sie Aktualisieren aus.

AWS CLI

Um den Kulanzzzeitraum für die Integritätsprüfung für eine bestehende Gruppe zu ändern

Fügen Sie die `--health-check-grace-period` Option dem [update-auto-scaling-group](#) Befehl hinzu. Im folgenden Beispiel wird die Übergangsfrist für die Integritätsprüfung mit einem Wert von **120** Sekunden für eine vorhandene Auto-Scaling-Gruppe mit dem Namen `my-asg` konfiguriert.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--health-check-grace-period 120
```

Note

Wir empfehlen dringend, auch die standardmäßige Instance-Vorbereitungs- und Wartefrist für Ihre Auto-Scaling-Gruppe festzulegen. Weitere Informationen finden Sie unter [Legen Sie die standardmäßige Instance-Vorbereitung für eine Auto-Scaling-Gruppe fest](#).

Überwachen Sie Auto Scaling Scaling-Instances mit beeinträchtigten Amazon EBS-Volumes mithilfe von Zustandsprüfungen

Sie können die Amazon EBS-Zustandsprüfungen für Ihre Auto Scaling-Gruppe aktivieren, um sicherzustellen, dass Amazon EC2 Auto Scaling das gesamte System überwacht, auf dem Ihre Anwendung ausgeführt wird.

Nachdem Sie diese Zustandsprüfungen aktiviert haben, empfängt Amazon EC2 Auto Scaling die Ergebnisse der Amazon EC2-Statusprüfungen, die an den angehängten EBS-Volumes einer Instance durchgeführt wurden. Wenn ein Volume nicht erreichbar ist oder die I/O Statusprüfungen nicht besteht, schlägt die Zustandsprüfung fehl und die entsprechende Instance wird als fehlerhaft eingestuft. Wenn Amazon EC2 Auto Scaling eine fehlerhafte Instance erkennt, wird sie ersetzt.

In diesem Thema wird davon ausgegangen, dass Sie mit den beigefügten EBS-Statusprüfungen vertraut sind. Falls nicht, finden Sie weitere Informationen im Abschnitt [Angehängte EBS-Statusprüfungen](#) im Amazon EC2 EC2-Benutzerhandbuch. Im folgenden Thema wird beschrieben, wie Sie die Amazon EC2 Auto Scaling Scaling-Zustandsprüfungen aktivieren, die auf den angehängten EBS-Statusprüfungen basieren.

Note

Sie können die Amazon EBS-Zustandsprüfungen für alle Ihre Auto Scaling Scaling-Gruppen aktivieren. Diese Integritätsprüfungen sind jedoch nur für [Instances verfügbar, die auf dem AWS Nitro-System basieren](#).

Aktivieren Sie die Amazon EBS-Zustandsprüfungen für eine Gruppe

Sie können die Amazon EBS-Zustandsprüfungen für neue und bestehende Auto Scaling Scaling-Gruppen aktivieren.

Console

Amazon EBS Health Checks für eine neue Gruppe aktivieren

Wenn Sie die Auto Scaling Scaling-Gruppe erstellen, wählen Sie auf der Seite Erweiterte Optionen konfigurieren für Integritätsprüfungen, Zusätzliche Zustandsprüfungstypen die Option Amazon EBS-Zustandsprüfungen aktivieren aus. Geben Sie dann unter Frist für

Zustandsprüfungen die Zeit in Sekunden ein. Diese Zeitspanne gibt an, wie lange Amazon EC2 Auto Scaling warten muss, bevor der Integritätsstatus einer Instance überprüft wird, nachdem sie den InService Status erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartefrist für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).

AWS CLI

Amazon EBS Health Checks für eine neue Gruppe aktivieren

Fügen Sie die `--health-check-type` Option dem [create-auto-scaling-group](#) Befehl hinzu. Das folgende Beispiel spezifiziert **EBS** für die `--health-check-type` Option für eine neue Auto Scaling Group mit dem Namen `my-asg`.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60 ...
```

Sie können mehrere Werte für die `--health-check-type` Option angeben. Verwenden Sie beispielsweise den folgenden Befehl, um sowohl Amazon EBS- als auch Elastic Load Balancing Balancing-Zustandsprüfungstypen hinzuzufügen.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS,ELB" --health-check-grace-period 60 ...
```

Bei den Wertnamen muss die Groß- und Kleinschreibung beachtet werden.

Console

Amazon EBS-Zustandsprüfungen für eine bestehende Gruppe aktivieren

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die AWS-Region aus, in der Sie Ihre Auto-Scaling-Gruppe erstellt haben.
3. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

4. Wählen Sie auf der Registerkarte Details die Option Zustandsprüfungen, Bearbeiten aus.

5. Wählen Sie für Integritätsprüfungen, Zusätzliche Zustandsprüfungsarten die Option Amazon EBS-Zustandsprüfungen aktivieren aus.
6. Geben Sie unter Frist für Zustandsprüfungen die Zeit in Sekunden ein. Diese Zeitspanne gibt an, wie lange Amazon EC2 Auto Scaling warten muss, bevor der Integritätsstatus einer Instance überprüft wird, nachdem sie den InService Status erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).
7. Wählen Sie Aktualisieren aus.

AWS CLI

Amazon EBS-Zustandsprüfungen für eine bestehende Gruppe aktivieren

Fügen Sie die `--health-check-type` Option dem [update-auto-scaling-group](#) Befehl hinzu. Im folgenden Beispiel wird **EBS** für die `--health-check-type` Option für eine bestehende Auto Scaling Group mit dem Namen angegeben `my-asg`.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60
```

Um mehrere Typen von Integritätsprüfungen zu verwenden, können Sie mehrere Werte (z. B. **EBS, ELB**) für die `--health-check-type` Option angeben.

Bei den Wertnamen muss die Groß- und Kleinschreibung beachtet werden.

Deaktivieren Sie die Amazon EBS-Zustandsprüfungen für eine Auto Scaling Group

Im folgenden Thema wird beschrieben, wie Sie die Amazon EBS-Zustandsprüfungen für eine Auto Scaling Group deaktivieren. Wenn Sie Amazon EBS Health Checks nicht mehr benötigen, gehen Sie wie folgt vor, um sie zu deaktivieren.

Console

Amazon EBS-Zustandsprüfungen für eine Gruppe ausschalten

1. Öffnen Sie die Amazon EC2 Console unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.

2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Zustandsprüfungen, Bearbeiten aus.
4. Deaktivieren Sie für Integritätsprüfungen, Zusätzliche Zustandsprüfungsarten die Option Amazon EBS-Zustandsprüfungen aktivieren.
5. Wählen Sie Aktualisieren aus.

AWS CLI

Amazon EBS-Zustandsprüfungen für eine Gruppe ausschalten

Verwenden Sie den [update-auto-scaling-group](#) Befehl, um die Integritätsprüfungen für eine Auto Scaling Group so zu aktualisieren, dass sie keine Amazon EBS-Zustandsprüfungen mehr verwendet. Geben Sie die Option `--health-check-type` und den Wert **EC2** ein.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--health-check-type "EC2"
```

Um Amazon EBS-Zustandsprüfungen zu deaktivieren, ohne andere Zustandsprüfungstypen (wie Elastic Load Balancing) zu deaktivieren, müssen Sie sie anstelle von **EC2** angeben. Geben Sie beispielsweise für Integritätsprüfungen von Elastic Load Balancing **ELB** die `--health-check-type` Option an.

Bei den Wertnamen muss die Groß- und Kleinschreibung beachtet werden.

Richten Sie eine benutzerdefinierte Integritätsprüfung für Ihre Auto Scaling Group ein

Sie können benutzerdefinierte Zustandsprüfungen verwenden, um die bestehenden Zustandsprüfungsoptionen von Amazon EC2 Auto Scaling zu ergänzen. Durch die Kombination von benutzerdefinierten Zustandsprüfungen mit den anderen Zustandsprüfungsarten können Sie ein umfassendes System zur Gesundheitsüberwachung erstellen, das auf die Bedürfnisse Ihrer Anwendung zugeschnitten ist.

Erstellen Sie zunächst benutzerdefinierte Tests, um zu überprüfen, ob die Instances in Ihrer Auto Scaling Group ordnungsgemäß funktionieren und eingehenden Datenverkehr verarbeiten

können. Wenn die von Ihnen konfigurierte Zustandsprüfung feststellt, dass eine Instance nicht reagiert, markieren Sie diese bestimmte Instance als `Unhealthy`, sodass Amazon EC2 Auto Scaling sie sofort ersetzt.

Sie können den Integritätsstatus einer Instance direkt an Amazon EC2 Auto Scaling senden, indem Sie das AWS CLI oder ein SDK verwenden. Die folgenden Beispiele zeigen Ihnen, wie Sie den AWS CLI Integritätsstatus einer Instance konfigurieren und anschließend den Integritätsstatus der Instance überprüfen können.

Verwenden Sie den folgenden [set-instance-health](#) Befehl, um den Integritätsstatus der angegebenen Instanz auf festzulegen `Unhealthy`.

```
aws autoscaling set-instance-health --instance-id i-1234567890abcdef0 --health-status Unhealthy
```

Standardmäßig wird bei diesem Befehl die Wartezeit für die Zustandsprüfung eingehalten. Sie können jedoch dieses Verhalten außer Kraft setzen und die Übergangszeit nicht einhalten, indem Sie die Option `--no-should-respect-grace-period` einbeziehen.

Verwenden Sie den folgenden [describe-auto-scaling-groups](#) Befehl, um zu überprüfen, ob der Integritätsstatus der Instanz lautet `Unhealthy`.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

Im Folgenden finden Sie eine Beispielantwort, die Ihnen zeigt, dass der Integritätsstatus der Instance lautet `Unhealthy` und dass die Instance beendet wird.

```
{
  "AutoScalingGroups": [
    {
      ....
      "Instances": [
        {
          "ProtectedFromScaleIn": false,
          "AvailabilityZone": "us-west-2a",
          "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1",
            "LaunchTemplateId": "lt-1234567890abcdef0"
          },
        },
      ],
    },
  ],
}
```

```
        "InstanceId": "i-1234567890abcdef0",
        "InstanceType": "t2.micro",
        "HealthStatus": "Unhealthy",
        "LifecycleState": "Terminating"
    },
    ...
]
}
}
```

Anzeigen des Grundes für Fehler bei Zustandsprüfung

Mithilfe des folgenden Verfahrens können Sie Informationen zu allen Instances einsehen, die aufgrund einer Zustandsprüfung ersetzt wurden.

Standardmäßig erstellt Amazon EC2 Auto Scaling eine neue Skalierung zum Beenden der fehlerhaften Instance und beendet diese anschließend. Während die Instance beendet wird, startet eine andere Skalierungsaktivität eine neue Instance. Sie können dieses Verhalten ändern, um so schnell wie möglich mit dem Start einer neuen Instance zu beginnen, indem Sie eine Instance-Wartungsrichtlinie verwenden. Weitere Informationen finden Sie unter [Wartungsrichtlinien für Instances](#).

Console

Den Grund für fehlgeschlagene Zustandsprüfungen anzeigen

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Auf der Registerkarte Activity (Aktivität) wird unter Activity history (Aktivitätsverlauf) in der Spalte Status angezeigt, ob Ihre Auto-Scaling-Gruppe Instances erfolgreich gestartet oder beendet hat.

Wenn es Instances fehlerhaft beendet hat, zeigt die Spalte Ursache das Datum und die Uhrzeit der Beendigung und den Grund für den Fehler der Zustandsprüfung an. Beispiel, At 2022-05-14T20:11:53Z an instance was taken out of service in response

to a user health-check. Diese Meldung weist darauf hin, dass eine benutzerdefinierte Integritätsprüfung die Instance als fehlerhaft markiert hat.

Hilfe bei fehlgeschlagenen Zustandsprüfungen finden Sie unter [Fehlerbehebung bei fehlerhaften Instances in Amazon EC2 Auto Scaling](#).

AWS CLI

Den Grund für fehlgeschlagene Integritätsprüfungen anzeigen

Verwenden Sie den folgenden [describe-scaling-activities](#)-Befehl.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

Im Folgenden finden Sie ein Antwortbeispiel, das den Grund für das Fehlschlagen der Integritätsprüfung Cause enthält.

```
{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in response to a user health-check.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\": \"subnet-5ea0c127\", \"Availability Zone\": \"us-west-2a\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:283179a2-f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
    ...
  ]
}
```

Eine Beschreibung der Felder in der Ausgabe finden Sie unter [Aktivität](#) in der Amazon EC2 Auto Scaling API-Referenz.

Um die Skalierungsaktivitäten nach dem Löschen der Auto Scaling Scaling-Gruppe zu beschreiben, fügen Sie dem [describe-scaling-activities](#) Befehl die `--include-deleted-groups` Option hinzu.

Fehlerbehebung bei fehlerhaften Instances in Amazon EC2 Auto Scaling

Im Folgenden finden Sie die von Amazon EC2 Auto Scaling zurückgegebenen Fehlermeldungen, die möglichen Ursachen und die Schritte, die Sie zur Behebung der Probleme ergreifen können.

Wie Sie eine Fehlermeldung abrufen, erfahren Sie unter [Anzeigen des Grundes für Fehler bei Zustandsprüfung](#).

Fehlermeldungen

- [Eine Instance wurde aufgrund eines Fehlers der EC2-Instance-Zustandsprüfung außer Betrieb genommen](#)
- [Eine Instance wurde als Reaktion auf eine EC2-Zustandsprüfung, die anzeigte, dass sie beendet oder angehalten wurde, außer Betrieb genommen.](#)
- [Eine Instance wurde aufgrund eines Fehlers der Zustandsprüfung des ELB-Systems außer Betrieb genommen](#)
- [Weitere Ressourcen](#)

Eine Instance wurde aufgrund eines Fehlers der EC2-Instance-Zustandsprüfung außer Betrieb genommen

Problem: Auto-Scaling-Instances lassen die Amazon EC2-Zustandsprüfungen fehlschlagen.

Ursache 1: Wenn es Probleme gibt, die dazu führen, dass Amazon EC2 die Instances in Ihrer Auto Scaling-Gruppe als beeinträchtigt betrachtet, ersetzt Amazon EC2 Auto Scaling die Instances im Rahmen seiner Zustandsprüfungen automatisch.

Lösung 1: Wenn eine Überprüfung des Instance-Status fehlschlägt, müssen Sie das Problem in der Regel selbst beheben, indem Sie Änderungen an der Instance-Konfiguration vornehmen, bis Ihre Anwendung keine Probleme mehr aufweist. Um dieses Problem zu beheben, führen Sie die folgenden Schritte aus:

1. Erstellen Sie manuell eine Amazon EC2-Instance, die nicht Teil der Auto-Scaling-Gruppe ist, und untersuchen Sie das Problem. Allgemeine Hilfe bei der Untersuchung beeinträchtigter Instances

- finden Sie unter [Problembehandlung bei Instances mit fehlgeschlagenen Statusprüfungen](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Nachdem Sie sich vergewissert haben, dass Ihre Instance erfolgreich gestartet wurde und in Ordnung ist, verteilen Sie eine neue, fehlerfreie Instanzkonfiguration an die Auto-Scaling-Gruppe.
 3. Löschen Sie die von Ihnen erstellte Instance, um zu verhindern, dass Ihr AWS -Konto weiter belastet wird.

Eine Instance wurde als Reaktion auf eine EC2-Zustandssprüfung, die anzeigte, dass sie beendet oder angehalten wurde, außer Betrieb genommen.

Problem: Auto-Scaling-Instances, die angehalten, neu gestartet oder beendet wurden, werden ersetzt.

Ursache 1: Ein Benutzer hat die Instance manuell angehalten, neu gestartet oder beendet.

Lösung 1: Wenn Sie die Instances in Ihrer Auto Scaling Scaling-Gruppe beenden oder neu starten müssen, empfehlen wir, die Instances zuerst in den Standby-Modus zu versetzen. Weitere Informationen finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#).

Ursache 2: Amazon EC2 Auto Scaling versucht, Spot-Instances zu ersetzen, nachdem der Amazon EC2 Spot-Services die Instances unterbricht, da der Spot-Preis über Ihren Höchstpreis steigt oder die Kapazität nicht mehr verfügbar ist.

Lösung 2: Es gibt keine Garantie, dass eine Spot-Instance existiert, um die Anforderung zu einem bestimmten Zeitpunkt zu erfüllen. Sie können die folgenden Schritte versuchen:

- Verwenden Sie einen höheren Spot-Höchstpreis (möglicherweise den On-Demand-Preis). Wenn Sie Ihren Höchstpreis höher setzen, ist die Chance höher, dass der Amazon EC2 Spot-Service Ihre erforderliche Kapazität startet und erhält.
- Erhöhen Sie die Anzahl der verschiedenen Kapazitätspools, über die Sie Instances starten können, indem Sie mehrere Instance-Typen in mehreren Availability Zones ausführen. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).
- Wenn Sie mehrere Instance-Typen verwenden, sollten Sie die Funktion „Kapazitätsausgleich“ aktivieren. Dies ist nützlich, wenn der Amazon EC2 Spot-Service versucht, eine neue Spot-Instance zu starten, bevor eine laufende Instance beendet wird. Weitere Informationen finden Sie unter [Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances](#).

Ursache 3: Mit Capacity Blocks beendet Amazon EC2 alle Instances, die noch laufen, 30 Minuten vor der Endzeit des Capacity Blocks. Diese abrupte Beendigung veranlasst Ihre Auto Scaling Scaling-Gruppe, neue Instances zu starten, um die gewünschte Kapazität aufrechtzuerhalten, auch wenn der Kapazitätsblock endet.

Lösung 3: Versuchen Sie Folgendes, um dieses Problem zu beheben:

- Verringern Sie die gewünschte Kapazität der Auto Scaling Scaling-Gruppe, um zu verhindern, dass sie versucht, neue Instances zu starten. Weitere Informationen finden Sie unter [Manuelle Skalierung für Amazon EC2 Auto Scaling](#).
- Stellen Sie sicher, dass Sie in Ihrer Auto Scaling Scaling-Gruppe 30 Minuten vor dem Ende des Kapazitätsblocks skalieren, damit dieser Fehler nicht häufig auftritt. Stellen Sie sicher, dass alle Lifecycle-Hooks 30 Minuten vor dem Ende des Kapazitätsblocks abgeschlossen sind. Weitere Informationen finden Sie unter [Capacity BlocksFür Machine-Learning-Workloads verwenden](#).

Eine Instance wurde aufgrund eines Fehlers der Zustandsprüfung des ELB-Systems außer Betrieb genommen

Problem: Auto-Scaling-Instances bestehen womöglich die EC2-Zustandsprüfungen. Jedoch können die Elastic Load Balancing-Zustandsprüfungen für die Zielgruppen oder Classic Load Balancers fehlschlagen, bei denen die Auto-Scaling-Gruppe registriert ist.

Ursache 1: Wenn sich Ihre Auto Scaling-Gruppe auf Zustandsprüfungen von Elastic Load Balancing stützt, bestimmt Amazon EC2 Auto Scaling den Integritätsstatus Ihrer Instances, indem es die Ergebnisse sowohl der EC2-Statusprüfungen als auch der Elastic Load Balancing Balancing-Zustandsprüfungen überprüft. Der Load Balancer führt Zustandsprüfungen durch, indem er eine Anforderung an jede Instance sendet und auf die richtige Antwort wartet, oder indem er eine Verbindung mit der Instance herstellt. Eine Instance kann bei einer Elastic Load Balancing-Zustandsprüfung fehlschlagen, weil eine Anwendung, die in der Instance ausgeführt wird, Probleme verursacht, die dazu führen, dass der Load Balancer die Instance außer Betrieb nimmt.

Lösung 1: So werden die Elastic Load Balancing-Zustandsprüfungen erfolgreich durchlaufen

- Stellen Sie sicher, dass die Einstellungen für die Zustandsprüfung Ihrer Zielgruppen korrekt konfiguriert sind. Sie definieren Zustandsprüfungseinstellungen für Ihren Load Balancer pro Zielgruppe. Weitere Informationen finden Sie unter [Konfigurieren Sie Integritätsprüfungen für Ziele](#).
- Überprüfen Sie die Erfolgscodes, die der Load Balancer erwartet, und stellen Sie sicher, dass Ihre Anwendung so konfiguriert ist, dass sie diese Codes bei Erfolg zurückgibt.

- Stellen Sie sicher, dass die Sicherheitsgruppen für Ihren Load Balancer und die Auto-Scaling-Gruppe korrekt konfiguriert sind.
- Stellen Sie sicher, dass der Load Balancer in denselben Availability Zones wie Ihre Auto-Scaling-Gruppe konfiguriert ist.

Lösung 2: Aktualisieren Sie die Auto Scaling Scaling-Gruppe, um die Integritätsprüfungen von Elastic Load Balancing zu deaktivieren. Anweisungen zum Deaktivieren dieser Zustandsprüfungen finden Sie unter [Eine Zielgruppe trennen oder unter Classic Load Balancer](#).

Ursache 2: Es besteht eine Diskrepanz zwischen dem Kulanzzzeitraum der Zustandsprüfung und der Startzeit der Instance.

Lösung 3: Bearbeiten Sie den Kulanzzzeitraum für die Integritätsprüfung für Ihre Auto Scaling Scaling-Gruppe. Legen Sie den Kulanzzzeitraum auf einen ausreichend langen Zeitraum fest, um die Anzahl der aufeinanderfolgenden erfolgreichen Zustandsprüfungen zu unterstützen, die erforderlich sind, bevor Elastic Load Balancing eine neu gestartete Instance als fehlerfrei einstuft. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).

Weitere Ressourcen

Wenn Sie ein anderes Problem haben, finden Sie in den folgenden AWS re:Post Artikeln zusätzliche Hilfe zur Fehlerbehebung:

- [Warum hat Amazon EC2 Auto Scaling eine Instance beendet?](#)
- [Warum hat Amazon EC2 Auto Scaling eine fehlerhafte Instance nicht beendet?](#)

Health Dashboard Benachrichtigungen für Amazon EC2 Auto Scaling

Ihr Health Dashboard bietet Unterstützung für Benachrichtigungen, die von Amazon EC2 Auto Scaling stammen. Diese Benachrichtigungen bieten umfassende Informationen sowie Anweisungen zur Behebung von Ressourcenleistungs- oder Verfügbarkeitsproblemen, die sich auf Ihre Anwendungen auswirken können. Derzeit sind nur Ereignisse verfügbar, die für fehlende Sicherheitsgruppen und Startvorlagen spezifisch sind.

Das Health Dashboard ist Teil des AWS Health Service. Sie benötigen keine Einrichtung und kann von jedem Benutzer angezeigt werden, der in Ihrem Konto authentifiziert ist. Weitere Informationen finden Sie unter [Erste Schritte mit Ihrem AWS Health Dashboard](#).

Wenn Sie eine Nachricht ähnlich den folgenden erhalten, sollte sie als Alarm behandelt werden, um Maßnahmen zu ergreifen.

Beispiel: Auto Scaling-Gruppe wird aufgrund einer fehlenden Sicherheitsgruppe nicht skaliert

Hello,

At 2020-01-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in AWS-Konto 123456789012.

A security group associated with this Auto Scaling group cannot be found. Each time a scale out operation is performed, it will be prevented until you make a change that fixes the issue.

We recommend that you review and update your Auto Scaling group configuration to change the launch template or launch configuration that depends on the unavailable security group.

Sincerely,
Amazon Web Services

Beispiel: Auto Scaling-Gruppe wird aufgrund einer fehlenden Startvorlage nicht skaliert

Hello,

At 2021-05-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in AWS-Konto 123456789012.

The launch template associated with this Auto Scaling group cannot be found. Each time a scale out operation is performed, it will be prevented until you make a change that fixes the issue.

We recommend that you review and update your Auto Scaling group configuration and specify an existing launch template to use.

Sincerely,
Amazon Web Services

Überwachen Sie CloudWatch Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances

Metriken sind das grundlegende Konzept bei Amazon CloudWatch. Eine Metrik steht für einen nach der Zeit geordneten Satz von Datenpunkten, die veröffentlicht werden. CloudWatch Sie können sich eine Metrik als eine zu überwachende Variable und die Datenpunkte als die Werte dieser Variablen im Laufe der Zeit vorstellen. Mit diesen Metriken können Sie überprüfen, ob Ihr System die erwartete Leistung zeigt.

Amazon EC2 Auto Scaling-Metriken, die Informationen zu Auto-Scaling-Gruppen sammeln, befinden sich im Namespace `AWS/AutoScaling`. Amazon EC2-Instance-Metriken, die CPU-Auslastungsdaten und andere Nutzungsdaten von Auto Scaling-Instances erfassen, befinden sich im Namespace `AWS/EC2`.

In der Konsole von Amazon EC2 Auto Scaling wird eine Reihe von Diagrammen für die Gruppenmetriken und die aggregierten Instance-Metriken für die Gruppe angezeigt. Je nach Ihren Anforderungen ziehen Sie es möglicherweise vor, auf Daten für Ihre Auto Scaling Scaling-Gruppen und -Instances von Amazon CloudWatch statt über die Amazon EC2 Auto Scaling Scaling-Konsole zuzugreifen.

Weitere Informationen finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).

Inhalt

- [Überwachungsgrafiken in der Amazon EC2 Auto Scaling-Konsole anzeigen](#)
- [CloudWatch Amazon-Metriken für Amazon EC2 Auto Scaling](#)
- [Überwachung für Auto-Scaling-Instances konfigurieren](#)

Überwachungsgrafiken in der Amazon EC2 Auto Scaling-Konsole anzeigen

Im Bereich Amazon EC2 Auto Scaling der Amazon EC2 EC2-Konsole können Sie den minute-by-minute Fortschritt einzelner Auto Scaling Scaling-Gruppen anhand CloudWatch von Metriken überwachen.

Sie können die folgenden Arten von Metriken überwachen:

- Auto Scaling-Metriken – Die Metriken für die automatische Skalierung werden nur aktiviert, wenn Sie sie einschalten. Weitere Informationen finden Sie unter [Aktivieren der Auto-Scaling-Metriken \(Konsole\)](#). Wenn die Auto-Scaling-Metriken aktiviert sind, zeigen die Überwachungsgrafiken Daten an, die mit einer Granularität von einer Minute für Auto-Scaling-Metriken veröffentlicht werden.
- EC2-Metriken – Die Amazon-EC2-Instance-Metriken sind immer aktiviert. Wenn die detaillierte Überwachung aktiviert ist, zeigen die Überwachungsdiagramme Daten, die mit einer Granularität von einer Minute für Instance-Metriken veröffentlicht werden. Weitere Informationen finden Sie unter [Überwachung für Auto-Scaling-Instances konfigurieren](#).

So zeigen Sie Überwachungsgrafiken über die Amazon EC2 Auto Scaling-Konsole an

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe, für die Sie Metriken anzeigen möchten.

Im unteren Teil der Seite Auto-Scaling-Gruppen wird ein geteilter Bereich geöffnet.

3. Wählen Sie die Registerkarte Überwachung.

Amazon EC2 Auto Scaling zeigt Überwachungsgrafiken für Auto-Scaling-Metriken an.

4. Wenn Überwachungsdiagramme der aggregierten Instance-Metriken für die Gruppe angezeigt werden sollen, wählen Sie EC2 aus.

Graph-Aktionen

- Bewegen Sie den Mauszeiger über einen Datenpunkt, um ein Daten-Popup für eine bestimmte Zeit in UTC anzuzeigen.
- Um ein Diagramm zu vergrößern, wählen Sie Vergrößern aus dem Menüwerkzeug (die drei vertikalen Punkte) oben rechts im Diagramm. Alternativ können Sie auch auf das Symbol zum Maximieren am oberen Rand des Diagramms klicken.
- Passen Sie den Zeitraum für die im Diagramm angezeigten Daten an, indem Sie einen der vordefinierten Werte für den Zeitraum auswählen. Wenn das Diagramm vergrößert ist, können Sie Benutzerdefiniert wählen, um Ihren eigenen Zeitraum zu definieren.
- Wählen Sie Aktualisieren aus dem Menü Werkzeug, um die Daten in einem Diagramm zu aktualisieren.

- Ziehen Sie den Mauszeiger über die Diagrammdaten, um einen bestimmten Bereich auszuwählen. Sie können dann im Menü Werkzeug die Option Zeitbereich anwenden wählen.
- Wählen Sie im Menü-Tool die Option Protokolle anzeigen, um die zugehörigen Log-Streams (falls vorhanden) in der CloudWatch Konsole anzuzeigen.
- Um ein Diagramm anzuzeigen CloudWatch, wählen Sie im Menü-Tool die Option In Metriken anzeigen aus. Dadurch gelangen Sie auf die CloudWatch Seite für das Diagramm. Dort können Sie weitere Informationen einsehen oder auf historische Daten zugreifen, um besser zu verstehen, wie sich Ihre Auto Scaling Gruppe über einen längeren Zeitraum verändert hat.

Graphische Metriken für Ihre Auto-Scaling-Gruppen

Nachdem Sie eine Auto-Scaling-Gruppe erstellt haben, können Sie die Amazon EC2 Auto Scaling-Konsole öffnen und die Überwachungsgraphen für die Gruppe auf der Registerkarte Überwachung anzeigen.

Im Abschnitt Auto Scaling enthalten die Diagramm-Metriken die folgenden Metriken. Diese Metriken liefern Messwerte, die Indikatoren für ein potenzielles Problem sein können, wie z.B. die Anzahl der abgebrochenen Instances oder die Anzahl der ausstehenden Instances. Sie finden Definitionen für diese Metriken unter [CloudWatch Amazon-Metriken für Amazon EC2 Auto Scaling](#).

Anzeigename	CloudWatch Name der Metrik
Minimale Gruppengröße	GroupMinSize
Maximale Gruppengröße	GroupMaxSize
Gewünschte Kapazität	GroupDesiredCapacity
In-Service-Instances	GroupInServiceInstances
Ausstehende Instances	GroupPendingInstances
Standby-Instances	GroupStandbyInstances
Beendende Instances	GroupTerminatingInstances
Gesamtzahl der Instances	GroupTotalInstances

Im Abschnitt EC2 finden Sie die folgenden grafischen Metriken, die auf den wichtigsten Leistungskennzahlen für Ihre Amazon EC2-Instanzen basieren. Diese EC2-Metriken sind ein Aggregat von Metriken für alle Instanzen in der Gruppe. Definitionen für diese Metriken finden Sie unter [Liste der verfügbaren CloudWatch Metriken für Ihre Instanzen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Anzeigename	CloudWatch Name der Metrik
CPU-Auslastung	CPUUtilization
Datenträgerlesevorgänge	DiskReadBytes
Festplattenlesevorgänge	DiskReadOps
Datenträgerschreibvorgänge	DiskWriteBytes
Festplattenschreibvorgänge	DiskWriteOps
Netzwerkeingang	NetworkIn
Netzwerkausgang	NetworkOut
Statusprüfung fehlgeschlagen (Beliebig)	StatusCheckFailed
Statusprüfung fehlgeschlagen (Instance)	StatusCheckFailed_Instance
Statusprüfung fehlgeschlagen (System)	StatusCheckFailed_System

Darüber hinaus sind einige Metriken für bestimmte Anwendungsfälle in den grafisch dargestellten Metriken für Auto Scaling verfügbar.

Die folgenden Metriken sind nützlich, wenn Ihre Gruppe Gewichtungen verwendet, die definieren, wie viele Einheiten jede Instance zur gewünschten Kapazität der Gruppe beiträgt. Sie finden Definitionen für diese Metriken unter [CloudWatch Amazon-Metriken für Amazon EC2 Auto Scaling](#).

Anzeigename	CloudWatch Name der Metrik
In-Service-Kapazitätseinheiten	GroupInServiceCapacity
Ausstehende Kapazitätseinheiten	GroupPendingCapacity
Standby-Kapazitätseinheiten	GroupStandbyCapacity
Beendende Kapazitätseinheiten	GroupTerminatingCapacity
Gesamte Kapazitätseinheiten	GroupTotalCapacity

Die folgenden Metriken sind nützlich, wenn Ihre Gruppe das Feature [Warmer Pool](#) verwendet. Sie finden Definitionen für diese Metriken unter [CloudWatch Amazon-Metriken für Amazon EC2 Auto Scaling](#).

Anzeigename	CloudWatch Name der Metrik
Mindestgröße des warmen Pools	WarmPoolMinSize
Gewünschte Kapazität des warmen Pools	WarmPoolDesiredCapacity
Ausstehende Kapazitätseinheiten des warmen Pools	WarmPoolPendingCapacity
Beenden der Kapazitätseinheiten des warmen Pools	WarmPoolTerminatingCapacity
Aufgewärmte Kapazitätseinheiten des warmen Pools	WarmPoolWarmedCapacity
Insgesamt gestartete Kapazitätseinheiten des warmen Pools	WarmPoolTotalCapacity

Anzeigename	CloudWatch Name der Metrik
Gewünschte Kapazität für Gruppe und warmen Pool	GroupAndWarmPoolDesiredCapacity
Gruppe und insgesamt gestartete Kapazitätseinheiten des warmen Pools	GroupAndWarmPoolTotalCapacity

Zugehörige Ressourcen

- Informationen zur Überwachung von Metriken pro Instanz finden Sie unter [Graph-Metriken für Ihre Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
- CloudWatch Dashboards sind anpassbare Homepages in der Konsole. CloudWatch Sie können diese Seiten verwenden, um die Ressourcen in einer Ansicht zu überwachen, auch Ressourcen, die über mehrere Regionen verteilt sind. Sie können CloudWatch Dashboards verwenden, um benutzerdefinierte Ansichten der Messwerte und Alarme für Ihre AWS Ressourcen zu erstellen. Weitere Informationen finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).

CloudWatch Amazon-Metriken für Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling veröffentlicht folgende Metriken im Namespace `AWS/AutoScaling`. Die tatsächlich verfügbaren Metriken für Auto-Scaling-Gruppen hängen davon ab, ob Sie Gruppenmetriken aktiviert haben und welche Gruppenmetriken Sie aktiviert haben. Gruppenmetriken stehen ohne zusätzliche Kosten mit einer Granularität von einer Minute zur Verfügung. Sie müssen sie jedoch aktivieren.

Wenn Sie Auto Scaling-Gruppenmetriken aktivieren, sendet Amazon EC2 Auto Scaling Stichprobendaten nach bestem Wissen und Gewissen an CloudWatch jede Minute. In seltenen Fällen, wenn es zu einer CloudWatch Serviceunterbrechung kommt, werden Daten nicht aufgefüllt, um Lücken im Verlauf der Gruppenmetriken zu schließen.

Inhalt

- [Metriken zu Auto-Scaling-Gruppen](#)
- [Dimensionen für Metriken zu Auto-Scaling-Gruppen](#)
- [Metriken und Dimensionen für die prädiktive Skalierung](#)

- [Aktivieren der Auto-Scaling-Metriken \(Konsole\)](#)
- [Aktivieren der Metriken zu Auto-Scaling-Gruppen \(AWS CLI\)](#)

Metriken zu Auto-Scaling-Gruppen

Mit diesen Metriken erhalten Sie nahezu kontinuierliche Einblicke in den Verlauf Ihrer Auto-Scaling-Gruppe. Hierzu zählen beispielsweise Größenänderungen der Gruppe im Laufe der Zeit.

Metrik	Description
GroupMinSize	<p>Die Mindestgröße der Auto-Scaling-Gruppe.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupMaxSize	<p>Die maximale Größe der Auto-Scaling-Gruppe.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupDesiredCapacity	<p>Die Anzahl von Instances, welche die Auto-Scaling-Gruppe beizubehalten versucht.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupInServiceInstances	<p>Die Anzahl von Instances, die im Rahmen der Auto-Scaling-Gruppe ausgeführt werden. Diese Metrik umfasst keine Instances, die schwebend oder beendet sind.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupPendingInstances	<p>Die Anzahl von schwebenden Instances. Eine schwebende Instance ist noch nicht in Betrieb. Diese Metrik umfasst keine Instances, die in Betrieb oder beendet sind.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>

Metrik	Description
GroupStandbyInstances	<p>Die Anzahl von Instances mit dem Status Standby. Instances in diesem Status werden zwar ausgeführt, sind aber nicht aktiv in Betrieb.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupTerminatingInstances	<p>Die Anzahl von Instances, die beendet werden. Diese Metrik beinhaltet keine Instances, die in Betrieb sind, ausstehen oder nach der Auto Scaling Scaling-Gruppe in einen warmen Pool zurückkehren.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupTerminatingRetainedInstances	<p>Die Anzahl der Instances im Terminating:Retained Bundesstaat.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupTotalInstances	<p>Die Gesamtanzahl von Instances in der Auto-Scaling-Gruppe. Diese Metrik gibt die Anzahl von Instances an, die in Betrieb, schwebend oder beendet sind.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>

Wenn Sie eine gemischte Instance-Gruppe so konfigurieren, dass die gewünschte Kapazität in verschiedenen Einheiten gemessen wird, z. B. indem Sie Gewichtungen auf der Grundlage der vCPU-Anzahl jedes Instance-Typs zuweisen, zählen die folgenden Metriken die Anzahl der von Ihrer Auto-Scaling-Gruppe verwendeten Einheiten. Wenn Sie eine gemischte Instance-Gruppe nicht so konfiguriert haben, dass die gewünschte Kapazität in verschiedenen Einheiten gemessen wird, werden die folgenden Metriken zwar ausgefüllt, entsprechen aber den in der vorherigen Tabelle definierten Metriken. Weitere Informationen finden Sie unter [Übersicht über die Einrichtung für die Erstellung einer gemischten Instance-Gruppe](#).

Metrik	Description
GroupInServiceCapacity	<p>Die Anzahl der Kapazitätseinheiten, die als Teil der Auto-Scaling-Gruppe ausgeführt werden.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupPendingCapacity	<p>Die Anzahl der schwebenden Kapazitätseinheiten.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupStandbyCapacity	<p>Die Anzahl der Kapazitätseinheiten, die sich in einem Standby-Status befinden.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupTerminatingCapacity	<p>Die Anzahl der Kapazitätseinheiten, die im Begriff sind, beendet zu werden.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupTerminatingRetainedCapacity	<p>Die Kapazitätseinheiten, die durch Instances im Terminating:Retained Bundesstaat repräsentiert werden.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupTotalCapacity	<p>Die Gesamtanzahl der Kapazitätseinheiten in der Auto-Scaling-Gruppe.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>

Für Auto-Scaling-Gruppen mit warmem Pool meldet Amazon EC2 Auto Scaling zudem folgende Metriken. Weitere Informationen finden Sie unter [Reduzieren Sie die Latenz für Anwendungen mit langen Startzeiten mithilfe warmer Pools](#).

Metrik	Description
WarmPoolMinSize	<p>Die Mindestgröße des warmen Pools.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
WarmPoolDesiredCapacity	<p>Die Menge an Kapazität, die Amazon EC2 Auto Scaling im warmen Pool zu erhalten versucht.</p> <p>Dies entspricht der maximalen Größe der Auto-Scaling-Gruppe abzüglich ihrer gewünschten Kapazität oder, falls eingestellt, der maximalen vorbereiteten Kapazität der Auto-Scaling-Gruppe abzüglich ihrer gewünschten Kapazität.</p> <p>Wenn jedoch die Mindestgröße des warmen Pools gleich oder größer ist als die Differenz zwischen der maximalen Größe (oder, falls eingestellt, der maximal vorbereiteten Kapazität) und der gewünschten Kapazität der Auto-Scaling-Gruppe, dann entspricht die gewünschte Kapazität des warmen Pools der WarmPoolMinSize .</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
WarmPoolPendingCapacity	<p>Die Menge an Kapazität im warmen Pool, die noch ausstehend ist. Dazu gehören Instances, die nach der Auto Scaling Scaling-Gruppe in einen warmen Pool zurückkehren. Diese Metrik umfasst keine laufenden, gestoppten oder beendeten Instances.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
WarmPoolPendingRetainedCapacity	<p>Verfolgt beibehaltene Instanzen, die in den warmen Pool zurückkehren.</p>

Metrik	Description
	Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.
WarmPoolTerminationCapacity	<p>Die Menge der Kapazität im warmen Pool, die gerade abgebaut wird. Diese Metrik umfasst keine laufenden, gestoppten oder ausstehenden Instances.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
WarmPoolTerminationRetainedCapacity	<p>Verfolgt beibehaltene Instances, die aus dem warmen Pool enden.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
WarmPoolWarmCapacity	<p>Die verfügbare Kapazität, um die Auto-Scaling-Gruppe während der Aufskalierung zu betreten. Diese Metrik umfasst keine Instances, die schwebend oder beendet sind.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
WarmPoolTotalCapacity	<p>Die Gesamtkapazität des warmen Pools, einschließlich der laufenden, gestoppten, anstehenden oder beendeten Instances.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>
GroupAndWarmPoolDesiredCapacity	<p>Die gewünschte Kapazität der Auto-Scaling-Gruppe und des warmen Pools zusammen.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>

Metrik	Description
GroupAndWarmPoolTotalCapacity	<p>Die Gesamtkapazität der Auto-Scaling-Gruppe und des warmen Pools zusammen. Dazu gehören laufende, gestoppte, ausstehende, beendete oder in Betrieb befindliche Instances.</p> <p>Reporting criteria (Berichtskriterien): Wird gemeldet, wenn die Metrikensammlung aktiviert ist.</p>

Dimensionen für Metriken zu Auto-Scaling-Gruppen

Sie können die folgenden Dimensionen verwenden, um die in den vorherigen Tabellen aufgeführten Metriken zu verfeinern.

Dimension	Description
AutoScalingGroupName	Filtert nach dem Namen einer Auto-Scaling-Gruppe.

Metriken und Dimensionen für die prädiktive Skalierung

Der Namespace `AWS/AutoScaling` enthält folgende Metriken für die prädiktive Skalierung.

Metriken sind mit einer Auflösung von einer Stunde verfügbar.

Zur Bewertung der Prognosegenauigkeit können Sie die prognostizierten Werte mit tatsächlichen Werten vergleichen. Weitere Informationen zur Bewertung der Prognosegenauigkeit mit diesen Metriken finden Sie unter [Überwachen Sie prädiktive Skalierungsmetriken mit CloudWatch](#).

Metrik	Description	Dimensionen
PredictiveScalingLoadForecast	<p>Die Last, die voraussichtlich von Ihrer Anwendung generiert wird.</p> <p>Die Statistiken Average, Minimum und Maximum sind hilfreich, die Statistik Sum allerdings nicht.</p>	AutoScalingGroupName , PolicyName , PairIndex

Metrik	Description	Dimensionen
	Berichtskriterien: Werden nach Erstellung der ursprüngliche Prognose gemeldet.	
PredictiveScalingCapacityForecast	<p>Die voraussichtliche Kapazität, die zur Deckung des Anwendungsbedarfs erforderlich ist. Dieser Wert basiert auf der Lastprognose und der gewünschten Zielauslastung für Ihre Auto Scaling-Instances.</p> <p>Die Statistiken Average, Minimum und Maximum sind hilfreich, die Statistik Sum allerdings nicht.</p> <p>Berichtskriterien: Werden nach Erstellung der ursprüngliche Prognose gemeldet.</p>	AutoScalingGroupName , PolicyName
PredictiveScalingMetricPairCorrelation	<p>Die Korrelation zwischen der Skalierungsmetrik und dem Durchschnitt der Lastmetrik pro Instance. Prädiktive Skalierung geht immer von einer hohen Korrelation aus. Wenn Sie also einen niedrigen Wert für diese Metrik beobachten, ist es besser, kein Metrikpaar zu verwenden.</p> <p>Die Statistiken Average, Minimum und Maximum sind hilfreich, die Statistik Sum allerdings nicht.</p> <p>Berichtskriterien: Werden nach Erstellung der ursprüngliche Prognose gemeldet.</p>	AutoScalingGroupName , PolicyName , PairIndex

Note

Die Dimension `PairIndex` gibt Informationen im Zusammenhang mit dem Index des Metrikpaars für Last und Skalierung gemäß Zuweisung durch Amazon EC2 Auto Scaling zurück. Der einzige gültige Wert ist derzeit `0`.

Aktivieren der Auto-Scaling-Metriken (Konsole)

So aktivieren Sie Metriken zu Gruppen

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Überwachung Auto-Scaling-Metriken aus und markieren Sie das Feld Aktivieren oben auf der Seite unter Auto Scaling.

So deaktivieren Sie Metriken zu Gruppen

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Ihre Auto-Scaling-Gruppe aus.
3. Entfernen Sie auf der Registerkarte Überwachung unter Erfassung von Metriken zu Auto-Scaling-Gruppen die Option Aktivieren.

Aktivieren der Metriken zu Auto-Scaling-Gruppen (AWS CLI)

Um Metriken für Auto-Scaling-Gruppen zu aktivieren

Aktivieren Sie eine oder mehrere Gruppenmetriken mithilfe des `enable-metrics-collection` Befehls. Mit dem folgenden Befehl wird beispielsweise eine einzelne Metrik für die angegebene Auto-Scaling-Gruppe aktiviert.

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \  
--metrics GroupDesiredCapacity --granularity "1Minute"
```

Wenn Sie die Option `--metrics` weglassen, werden alle Metriken aktiviert.

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \  
--granularity "1Minute"
```

Um Metriken für Auto-Scaling-Gruppen zu deaktivieren

Verwenden Sie den [disable-metrics-collection](#)Befehl, um alle Gruppenmetriken zu deaktivieren.

```
aws autoscaling disable-metrics-collection --auto-scaling-group-name my-asg
```

Überwachung für Auto-Scaling-Instances konfigurieren

Amazon EC2 sammelt und verarbeitet Rohdaten von Instances, um lesbare Quasi-Echtzeit-Metriken zu erhalten, die Aufschluss über die CPU-Auslastung und andere Nutzungsdaten für Ihre Auto-Scaling-Gruppe geben. Sie können das Intervall für die Überwachung dieser Metriken konfigurieren, indem Sie eine Granularität von einer Minute oder fünf Minuten auswählen.

Die Instance-Überwachung wird beim Start einer Instance aktiviert und es wird entweder die Basisüberwachung (mit einer Granularität von fünf Minuten) oder die detaillierte Überwachung (mit einer Granularität von einer Minute) verwendet. Für die detaillierte Überwachung werden zusätzliche Gebühren fällig. Weitere Informationen finden Sie unter [CloudWatch Amazon-Preise](#) und [Überwachung Ihrer Instances mithilfe von](#) Amazon CloudWatch im Amazon EC2-Benutzerhandbuch.

Es empfiehlt sich, vor der Erstellung einer Auto-Scaling-Gruppe eine Startvorlage oder eine Startkonfiguration zu erstellen, die den für Ihre Anwendung geeigneten Überwachungstyp zulässt. Wenn Sie Ihrer Gruppe eine Skalierungsrichtlinie hinzufügen, empfehlen wir dringend, die detaillierte Überwachung zu verwenden, um Metrikdaten für EC2-Instances im Minutentakt zu erhalten, da so schneller auf Laständerungen reagiert werden kann.

Inhalt

- [Aktivieren der detaillierten Überwachung \(Konsole\)](#)
- [Aktivieren der detaillierten Überwachung \(AWS CLI\)](#)
- [Zwischen grundlegender und detaillierter Überwachung wechseln](#)
- [Erfassen Sie mit dem CloudWatch Agenten zusätzliche Metriken](#)

Aktivieren der detaillierten Überwachung (Konsole)

Standardmäßig ist die grundlegende Überwachung aktiviert, wenn Sie die verwenden AWS-Managementkonsole , um eine Startvorlage oder eine Startkonfiguration zu erstellen.

Aktivieren der detaillierten Überwachung in einer Startvorlage

Wenn Sie die Startvorlage mithilfe von erstellen AWS-Managementkonsole, wählen Sie im Abschnitt Erweiterte Details für detaillierte CloudWatch Überwachung die Option Aktivieren aus. Andernfalls ist die grundlegende Überwachung aktiviert. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage mithilfe erweiterter Einstellungen](#).

Aktivieren der detaillierten Überwachung in einer Startvorlage

Wenn Sie die Startkonfiguration mithilfe von erstellen AWS-Managementkonsole, wählen Sie im Abschnitt Zusätzliche Konfiguration die Option Detaillierte Überwachung der EC2-Instance aktivieren aus. CloudWatch Andernfalls ist die grundlegende Überwachung aktiviert. Weitere Informationen finden Sie unter [Erstellen einer Startkonfiguration](#).

Aktivieren der detaillierten Überwachung (AWS CLI)

Standardmäßig ist die grundlegende Überwachung aktiviert, wenn Sie die AWS CLI verwenden, um eine Startvorlage zu erstellen. Die detaillierte Überwachung ist standardmäßig aktiviert, wenn Sie eine Startkonfiguration mithilfe der AWS CLI erstellen.

Aktivieren der detaillierten Überwachung in einer Startvorlage

Verwenden Sie für Startvorlagen den Befehl [create-launch-template](#) und übergeben Sie eine JSON-Datei, die die Informationen für die Erstellung der Startvorlage enthält. Legen Sie die Überwachungsparameter zum Aktivieren der detaillierten Überwachung auf "Monitoring": {"Enabled": true} und zum Aktivieren der grundlegenden Überwachung auf "Monitoring": {"Enabled": false} fest.

Aktivieren der detaillierten Überwachung in einer Startvorlage

Verwenden Sie für Startkonfigurationen den Befehl [create-launch-configuration](#) mit der Option --instance-monitoring. Legen Sie für diese Option zum Aktivieren der detaillierten Überwachung true oder zum Aktivieren der grundlegenden Überwachung false fest.

```
--instance-monitoring Enabled=true
```

Zwischen grundlegender und detaillierter Überwachung wechseln

Um die Art der Überwachung zu ändern, die auf neuen EC2-Instances aktiviert ist, aktualisieren Sie die Startvorlage oder aktualisieren Sie die Auto-Scaling-Gruppe so, dass sie eine neue Startvorlage oder Startkonfiguration verwendet. Für bestehende Instances wird weiterhin die zuvor aktivierte Überwachungsart verwendet. Um alle Instances zu aktualisieren, beenden Sie sie, damit sie durch Ihre Auto-Scaling Gruppe ersetzt werden, oder aktualisieren Sie Instances einzeln mithilfe des Befehls [monitor-instances](#) und des Befehls [unmonitor-instances](#).

Note

Mit den Funktionen Instance Refresh und maximale Instance-Lebensdauer können Sie auch alle Instances in der Auto-Scaling-Gruppe ersetzen, um neue Instances zu starten, welche die neuen Einstellungen verwenden. Weitere Informationen finden Sie unter [Ersetzen Sie die Instances in Ihrer Auto Scaling Scaling-Gruppe](#).

Wenn Sie zwischen grundlegender und detaillierter Überwachung wechseln:

Wenn Sie CloudWatch Alarme mit den schrittweisen Skalierungsrichtlinien oder einfachen Skalierungsrichtlinien für Ihre Auto Scaling Scaling-Gruppe verknüpft haben, verwenden Sie den [put-metric-alarm](#) Befehl, um jeden Alarm zu aktualisieren. Sorgen Sie dafür, dass jeder Zeitraum mit der Überwachungsart übereinstimmt (300 Sekunden bei der grundlegenden und 60 Sekunden bei der detaillierten Überwachung). Wenn Sie von der detaillierten zur grundlegenden Überwachung wechseln, den Zeitraum der Alarme jedoch nicht zu fünf Minuten ändern, werden weiterhin minütlich Statistiken abgerufen. In diesem Fall kann das Ergebnis in vier von fünf Zeitabschnitten lauten, dass keine Daten verfügbar sind.

Erfassen Sie mit dem CloudWatch Agenten zusätzliche Metriken

Um Metriken auf Betriebssystemebene wie verfügbaren und belegten Arbeitsspeicher zu erfassen, müssen Sie den CloudWatch Agenten installieren. Es können zusätzliche Gebühren anfallen. Sie können den CloudWatch Agenten verwenden, um sowohl Systemmetriken als auch Protokolldateien von Amazon EC2 EC2-Instances zu sammeln. Weitere Informationen finden Sie unter [Vom CloudWatch Agenten erhobene Metriken](#) im CloudWatch Amazon-Benutzerhandbuch.

Protokollieren Sie Amazon EC2 Auto Scaling API-Aufrufe mit AWS CloudTrail

Amazon EC2 Auto Scaling ist in einen Service integriert [AWS CloudTrail](#), der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem AWS-Service ausgeführten Aktionen bereitstellt. CloudTrail erfasst API-Aufrufe für Auto Scaling als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von AWS-Managementkonsole und Codeaufrufen zu den Auto Scaling Scaling-API-Vorgängen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Auto Scaling gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, den Zeitpunkt der Anfrage und weitere Details ermitteln.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anforderung mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Die Anforderung wurde im Namen eines IAM-Identity-Center-Benutzers erstellt.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Event-Verlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter [Arbeiten mit dem CloudTrail Ereignisverlauf](#). Für die Anzeige des Eventverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail.

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS-Managementkonsole sind regionsübergreifend. Sie können mithilfe von AWS CLI einen Einzel-Region- oder einen Multi-Region-Trail erstellen. Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten AWS-

Regionen in Ihrem Konto erfassen. Wenn Sie einen Einzel-Region-Trail erstellen, können Sie nur die Ereignisse anzeigen, die im AWS-Region des Trails protokolliert wurden. Weitere Informationen zu Trails finden Sie unter [Erstellen eines Trails für Ihr AWS-Konto](#) und [Erstellen eines Trails für eine Organisation](#) im AWS CloudTrail -Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter [AWS CloudTrail Preise](#). Informationen zu Amazon-S3-Preisen finden Sie unter [Amazon S3 – Preise](#).

Auto Scaling Scaling-Verwaltungsereignisse in CloudTrail

[Verwaltungsereignisse](#) liefern Informationen über Verwaltungsvorgänge, die an Ressourcen in Ihrem ausgeführt werden AWS-Konto. Sie werden auch als Vorgänge auf Steuerebene bezeichnet. CloudTrail Protokolliert standardmäßig Verwaltungsereignisse.

Amazon EC2 Auto Scaling protokolliert alle Operationen der Auto Scaling-Steuerungsebene als Verwaltungsereignisse. Eine Liste der Vorgänge auf der Amazon EC2 Auto Scaling-Steuerungsebene, bei denen Auto Scaling protokolliert wird CloudTrail, finden Sie in der [Amazon EC2 Auto Scaling API-Referenz](#).

Beispiele Auto Scaling Scaling-Ereignisse

Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über den angeforderten API-Vorgang, Datum und Uhrzeit des Vorgangs, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass Ereignisse nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das den `CreateLaunchConfiguration` Vorgang demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "Root",
    "principalId": "123456789012",
    "arn": "arn:aws:iam::123456789012:root",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
```

```
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2018-08-21T17:05:42Z"
        }
    },
    "eventTime": "2018-08-21T17:07:49Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "CreateLaunchConfiguration",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Coral/Jakarta",
    "requestParameters": {
        "ebsOptimized": false,
        "instanceMonitoring": {
            "enabled": false
        },
        "instanceType": "t2.micro",
        "keyName": "EC2-key-pair-oregon",
        "blockDeviceMappings": [
            {
                "deviceName": "/dev/xvda",
                "ebs": {
                    "deleteOnTermination": true,
                    "volumeSize": 8,
                    "snapshotId": "snap-01676e0a2c3c7de9e",
                    "volumeType": "gp2"
                }
            }
        ],
        "launchConfigurationName": "launch_configuration_1",
        "imageId": "ami-6cd6f714d79675a5",
        "securityGroups": [
            "sg-00c429965fd921483"
        ]
    },
    "responseElements": null,
    "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
    "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Informationen zu CloudTrail Datensatzinhalten finden Sie im AWS CloudTrail Benutzerhandbuch unter [CloudTrailDatensatzinhalt](#).

Auto Scaling RemoveAction ruft auf CloudWatch

In Ihrem AWS CloudTrail Protokoll wird möglicherweise angezeigt, dass Auto Scaling die CloudWatch RemoveAction API aufruft, wenn Auto Scaling anweist CloudWatch, die automatische Skalierungsaktion aus einem Alarm zu entfernen. Dies kann passieren, wenn Sie ein skalierbares Ziel abmelden, eine Skalierungsrichtlinie löschen oder wenn ein Alarm eine nicht existierende Skalierungsrichtlinie aufruft.

Amazon SNS SNS-Benachrichtigungsoptionen für Amazon EC2 Auto Scaling

Sie können Ihre Auto Scaling Scaling-Gruppe so konfigurieren, dass Sie über wichtige Ereignisse informiert werden, die sich auf Ihre Anwendung auswirken. Mithilfe von Benachrichtigungen können Sie auch Abfragen verhindern, sodass Sie nicht auf den RequestLimitExceeded Fehler stoßen, der manchmal bei Abfragen auftritt.

Es gibt zwei Möglichkeiten, Benachrichtigungen über Amazon EC2 Auto Scaling zu erhalten:

- Amazon Simple Notification Service — Amazon SNS kann Sie benachrichtigen, wenn Ihre Auto Scaling Scaling-Gruppe Instances startet oder beendet. Sie können Amazon SNS-Benachrichtigungen lediglich aktivieren oder deaktivieren. Weitere Informationen finden Sie unter [Amazon SNS und Amazon EC2 Auto Scaling](#).
- Amazon EventBridge — EventBridge bietet erweiterte, ereignisgesteuerte Benachrichtigungen, die bestimmten Kriterien entsprechen und an eine Vielzahl von Zielen gesendet werden, einschließlich Amazon SNS. EventBridge kann auch ein breiteres Spektrum von Auto Scaling Scaling-Ereignissen überwachen, um eine genauere Überwachung zu ermöglichen. Weitere Informationen finden Sie unter [Wird EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen verwendet](#).

Sie können optional Benachrichtigungen mit Lifecycle-Hooks verwenden, um beim Starten oder Beenden benutzerdefinierte Aktionen für Instances durchzuführen. Weitere Informationen zur Konfiguration der Benachrichtigungen für die Verwendung mit Lifecycle-Hooks finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).

Amazon SNS und Amazon EC2 Auto Scaling

In diesem Abschnitt wird gezeigt, wie Sie Amazon SNS verwenden, um zu überwachen, wann Ihre Auto Scaling Scaling-Gruppe Instances startet oder beendet.

Wenn Sie zum Beispiel Ihre Auto Scaling-Gruppe konfigurieren, die `autoscaling:EC2_INSTANCE_TERMINATE`-Benachrichtigungsweise zu benutzen, und Ihre Auto Scaling-Gruppe beendet eine Instance, sendet es eine Benachrichtigung per Mail. Diese E-Mail enthält die Details der beendeten Instance, z. B. die Instance-ID und den Grund für die Beendigung.

Beachten Sie, dass, wenn Amazon EC2 Auto Scaling Instances zur Gruppe hinzufügt oder daraus entfernt, Benachrichtigungen über diese Änderungen an Sie gesendet werden, wobei eine Benachrichtigung pro Instance gesendet wird. Die Zustellung dieser Benachrichtigungen erfolgt jedoch nach bestem Bemühen, und Ihre Instances können auch nach der ersten Benachrichtigung ausfallen, wenn beispielsweise eine spätere Zustandsprüfung fehlschlägt. Weitere Informationen zum Integritätsprüfungsprozess finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Weitere Informationen zu Amazon SNS im Allgemeinen finden Sie im [Amazon Simple Notification Service Developer Guide](#).

Inhalt

- [SNS-Benachrichtigungen](#)
- [Konfigurieren von Amazon SNS-Benachrichtigungen für Amazon EC2 Auto Scaling](#)
 - [Erstellen Sie ein Amazon SNS-Thema.](#)
 - [Amazon SNS-Thema abonnieren](#)
 - [Bestätigen Ihres Amazon SNS-Abonnements](#)
 - [Konfigurieren Ihrer Auto Scaling-Gruppe zum Senden von Benachrichtigungen](#)
 - [Testen der Benachrichtigung](#)
 - [Löschen der Benachrichtigungskonfiguration](#)
- [Schlüsselrichtlinie für ein verschlüsseltes Amazon-SNS-Thema](#)

SNS-Benachrichtigungen

Amazon EC2 Auto Scaling unterstützt das Senden von Amazon SNS-Benachrichtigungen bei folgenden Ereignissen.

Veranstaltung	Description
autoscaling:EC2_INSTANCE_LAUNCH	Erfolgreiches Starten einer Instance
autoscaling:EC2_INSTANCE_LAUNCH_ERROR	Fehlgeschlagenes Starten einer Instance
autoscaling:EC2_INSTANCE_TERMINATE	Erfolgreiches Beenden einer Instance
autoscaling:EC2_INSTANCE_TERMINATE_ERROR	Fehlgeschlagenes Beenden einer Instance

Die Nachricht enthält die folgenden Informationen:

- Event – Das Ereignis
- AccountId – Die Konto-ID von Amazon Web Services.
- AutoScalingGroupName – Der Name der Auto Scaling-Gruppe.
- AutoScalingGroupARN – Der ARN der Auto Scaling-Gruppe.
- EC2InstanceId – Die ID der EC2-Instance.

Beispiel:

```
Service: AWS Auto Scaling
Time: 2016-09-30T19:00:36.414Z
RequestId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:region:123456789012:autoScalingGroup...
ActivityId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Description: Launching a new EC2 instance: i-0598c7d356eba48d7
Cause: At 2016-09-30T18:59:38Z a user request update of AutoScalingGroup constraints
to ...
StartTime: 2016-09-30T19:00:04.445Z
EndTime: 2016-09-30T19:00:36.414Z
StatusCode: InProgress
StatusMessage:
```

```
Progress: 50
EC2InstanceId: i-0598c7d356eba48d7
Details: {"Subnet ID":"subnet-id","Availability Zone":"zone"}
Origin: AutoScalingGroup
Destination: EC2
```

Konfigurieren von Amazon SNS-Benachrichtigungen für Amazon EC2 Auto Scaling

Damit Sie Amazon SNS zum Versenden von E-Mail-Benachrichtigungen verwenden können, müssen Sie zunächst ein Thema erstellen und es mit Ihren E-Mail-Adressen abonnieren.

Erstellen Sie ein Amazon SNS-Thema.

Ein SNS-Thema ist ein logischer Zugriffspunkt, ein Kommunikationskanal der Auto Scaling-Gruppe zum Versenden von Benachrichtigungen. Sie erstellen ein Thema, indem Sie einen Namen dafür angeben.

Wenn Sie einen Themanamen vergeben, muss der Name folgende Anforderungen erfüllen:

- Er muss zwischen 1 und 256 Zeichen lang sein.
- Er muss ASCII-Buchstaben mit Groß- und Kleinschreibung, Zahlen, Unterstriche oder Bindestriche enthalten.

Weitere Informationen finden Sie unter [Amazon SNS-Thema anlegen](#) im Amazon Simple Notification Service-Entwicklerhandbuch.

Amazon SNS-Thema abonnieren

Zum Empfangen der Benachrichtigungen, die die Auto-Scaling-Gruppe an das Thema sendet, müssen Sie einen Endpunkt für das Thema abonnieren. In diesem Verfahren, für Endpoint, geben Sie die E-Mailadresse an, unter der Sie die Benachrichtigungen von Amazon EC2 Auto Scaling erhalten möchten.

Weitere Informationen finden Sie unter [Amazon SNS-Thema abonnieren](#) im Amazon Simple Notification Service-Entwicklerhandbuch.

Bestätigen Ihres Amazon SNS-Abonnements

Amazon SNS sendet eine Bestätigungs-E-Mail an die E-Mail-Adresse, die Sie im vorherigen Schritt angegeben haben.

Öffnen Sie die E-Mail von AWS Notifications und klicken Sie auf den Link zur Bestätigung des Abonnements, bevor Sie mit dem nächsten Schritt fortfahren.

Sie erhalten eine Bestätigungsnachricht von AWS Amazon SNS ist jetzt so konfiguriert, dass Benachrichtigungen empfangen und als E-Mail an die angegebene E-Mail-Adresse gesendet werden.

Konfigurieren Ihrer Auto Scaling-Gruppe zum Senden von Benachrichtigungen

Sie können die Auto Scaling-Gruppe so konfigurieren, dass im Falle von Skalierungsereignissen (z. B. Starten oder Beenden von Instances) Benachrichtigungen an Amazon SNS gesendet werden. Amazon SNS sendet eine Benachrichtigung mit Informationen zu den Instances an die E-Mail-Adresse, die Sie angegeben haben.

So konfigurieren Sie Amazon SNS-Benachrichtigungen für Ihre Auto Scaling-Gruppe (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet, in dem Informationen über die ausgewählte Gruppe angezeigt werden.

3. Klicken Sie auf der Registerkarte Aktivität Benachrichtigungen über Aktivitäten, Benachrichtigung erstellen.
4. Führen Sie im Bereich Create notifications die folgenden Schritte aus:
 - a. Wählen Sie unter SNS-Thema das SNS-Thema aus.
 - b. Wählen Sie unter Event types die Ereignisse aus, zu denen Benachrichtigungen gesendet werden sollen.
 - c. Wählen Sie Erstellen aus.

So konfigurieren Sie Amazon SNS-Benachrichtigungen für Ihre Auto Scaling-Gruppe (AWS CLI)

Verwenden Sie den folgenden [put-notification-configuration](#)-Befehl.

```
aws autoscaling put-notification-configuration --auto-scaling-group-name my-  
asg --topic-arn arn --notification-types "autoscaling:EC2_INSTANCE_LAUNCH"  
"autoscaling:EC2_INSTANCE_TERMINATE"
```

Testen der Benachrichtigung

Aktualisieren Sie die Auto Scaling-Gruppe, indem Sie die gewünschte Kapazität der Auto Scaling-Gruppe um 1 erhöhen, um eine Benachrichtigung für ein Startereignis zu generieren. Sie erhalten innerhalb weniger Minuten nach dem Start der Instance eine Benachrichtigung.

So ändern Sie die gewünschte Kapazität (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite Auto-Scaling-Gruppen wird ein geteilter Bereich geöffnet, in dem Informationen über die ausgewählte Gruppe angezeigt werden.

3. Wählen Sie auf der Registerkarte Details die Option Gruppendetails, Bearbeiten.
4. Erhöhen Sie für Desired capacity (Gewünschte Kapazität) den aktuellen Wert um 1. Wenn dieser Wert die Maximum capacity (Maximalkapazität) überschreitet, müssen Sie auch den Wert der Maximum capacity (Maximalkapazität) um 1 erhöhen.
5. Wählen Sie Aktualisieren aus.
6. Nach einigen Minuten erhalten Sie eine Benachrichtigung über das Ereignis. Wenn Sie die zusätzliche Instance, die Sie für diesen Test gestartet haben, nicht benötigen, können Sie Desired capacity (Gewünschte Kapazität) um 1 reduzieren. Nach einigen Minuten erhalten Sie eine Benachrichtigung über das Ereignis.

Löschen der Benachrichtigungskonfiguration

Sie können Ihre Amazon EC2 Auto Scaling-Benachrichtigungskonfiguration löschen, wenn sie nicht mehr verwendet wird.

So löschen Sie die Amazon EC2 Auto Scaling-Benachrichtigungskonfiguration (Konsole)

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Ihre Auto Scaling-Gruppe aus.
3. Klicken Sie auf der Registerkarte Aktivität auf das Kontrollkästchen neben der Benachrichtigung, die Sie löschen möchten, und wählen Sie dann Aktionen, Löschen aus.

So löschen Sie die Amazon EC2 Auto Scaling-Benachrichtigungskonfiguration (AWS CLI)

Verwenden Sie den folgenden delete-notification-configuration-Befehl.

```
aws autoscaling delete-notification-configuration --auto-scaling-group-name my-asg --  
topic-arn arn
```

Informationen über die Löschung des Amazon SNS -Themas und aller mit Ihrer Auto Scaling-Gruppe verbundenen Abonnements finden Sie unter [Löschen eines Amazon SNS-Abonnements und -Themas](#) im Amazon Simple Notification Service-Entwicklerhandbuch.

Schlüsselrichtlinie für ein verschlüsseltes Amazon-SNS-Thema

Das von Ihnen angegebene Amazon-SNS-Thema ist möglicherweise mit einem vom Kunden verwalteten Schlüssel verschlüsselt, der mit dem AWS Key Management Service erstellt wurde. Um Amazon EC2 Auto Scaling die Berechtigung zu erteilen, in verschlüsselten Themen zu veröffentlichen, müssen Sie zuerst Ihren KMS-Schlüssel erstellen und dann die folgende Anweisung zur Richtlinie für den KMS-Schlüssel hinzufügen. Ersetzen Sie den Beispiel-ARN durch den ARN der entsprechenden serviceverknüpften Rolle, der Zugriff auf den Schlüssel gewährt wird. Weitere Informationen erhalten Sie unter [AWS KMS -Berechtigungen konfigurieren](#) im Entwicklerhandbuch für Amazon Simple Notification Service.

In diesem Beispiel erteilt die Richtlinienerklärung der mit dem Service verknüpften Rolle namens `AWSServiceRoleForAutoScalingBerechtigungen` zur Verwendung des vom Kunden verwalteten Schlüssels. Weitere Informationen zur serviceverknüpften Rolle für Amazon EC2 Auto Scaling finden Sie unter [Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling](#).

```
{  
  "Sid": "Allow service-linked role use of the customer managed key",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam:123456789012:role/aws-service-role/autoscaling.amazonaws.com/  
AWSServiceRoleForAutoScaling"  
  },  
  "Action": [  
    "kms:GenerateDataKey*",  
    "kms:Decrypt"  
  ],  
  "Resource": "*"   
}
```

Die Bedingungsschlüssel `aws:SourceArn` und `aws:SourceAccount` werden in Schlüsselrichtlinien nicht unterstützt, die es Amazon EC2 Auto Scaling ermöglichen, in verschlüsselten Themen zu veröffentlichen.

AWS in Amazon EC2 Auto Scaling integrierte Services

Amazon EC2 Auto Scaling kann in andere AWS Services integriert werden. Lesen Sie die folgenden Integrationsoptionen, um mehr darüber zu erfahren, wie jeder Dienst mit Amazon EC2 Auto Scaling funktioniert.

Themen

- [Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances](#)
- [Reservieren Sie Kapazität in bestimmten Availability Zones mit Kapazitätsreservierungen](#)
- [Erstellen Sie Auto Scaling Scaling-Gruppen über die Befehlszeile mit AWS CloudShell](#)
- [Erstellen Sie Auto Scaling Scaling-Gruppen mit AWS CloudFormation](#)
- [Holen Sie sich Empfehlungen zum Instanztyp mit AWS Compute Optimizer](#)
- [Verwenden Sie Elastic Load Balancing, um den eingehenden Anwendungsdatenverkehr in Ihrer Auto Scaling Scaling-Gruppe zu verteilen](#)
- [Steuern Sie den Verkehrsfluss mit einer Zielgruppe von VPC Lattice](#)
- [Wird EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen verwendet](#)
- [Stellen Sie Netzwerkkonnektivität für Ihre Auto-Scaling-Instances mit Amazon VPC bereit](#)

Kapazitätsausgleich bei Auto Scaling als Ersatz für gefährdete Spot-Instances

Der Kapazitätsausgleich in Auto Scaling hilft Ihnen dabei, die Verfügbarkeit Ihrer Workloads aufrechtzuerhalten, indem Sie Spot-Instances, bei denen das Risiko einer Unterbrechung besteht, proaktiv ersetzen.

Wenn für Spot-Instances ein erhöhtes Risiko einer Unterbrechung besteht, sendet der Amazon EC2 Spot-Service eine Empfehlung zur Neuverteilung von EC2-Instances an Amazon EC2 Auto Scaling. Wenn Sie Capacity Rebalancing aktivieren, versucht Auto Scaling, proaktiv die Spot-Instances in Ihrer Gruppe zu ersetzen, die eine Empfehlung zur Neuverteilung von EC2-Instances erhalten haben. Dies gibt Ihnen die Möglichkeit Ihr Workload auf neue oder bestehende Spot-Instances auszugleichen, die nicht einem erhöhten Risiko einer Unterbrechung ausgesetzt sind.

Wenn Sie Capacity Rebalancing nicht verwenden, ersetzt Auto Scaling Spot-Instances erst, nachdem der Amazon EC2-Spot-Service die Instances unterbrochen hat und ihre Zustandsprüfung

fehlschlägt. Bevor Sie eine Instance unterbrechen, gibt Amazon EC2 immer sowohl eine EC2-Instance-Neuausgleichsempfehlung als auch zwei Minuten im Voraus einen Hinweis auf die Spot-Instance-Unterbrechung aus.

Inhalt

- [-Übersicht](#)
- [Verhalten bei Kapazitätswiederherstellungen](#)
- [Überlegungen](#)
- [Aktivieren Sie den Kapazitätsausgleich, um gefährdete Spot-Instances proaktiv zu ersetzen](#)

-Übersicht

Um Kapazitätswiederherstellungen mit Ihrer Auto-Scaling-Gruppe zu verwenden, gehen Sie wie folgt vor:

1. Konfigurieren Sie Ihre Auto-Scaling-Gruppe für die Verwendung mehrerer Instance-Typen und Availability Zones. Auf diese Weise kann Amazon EC2 Auto Scaling die verfügbare Kapazität für Spot-Instances in jeder Availability Zone betrachten. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).
2. Fügen Sie bei Bedarf Lebenszyklus-Hooks hinzu, um Ihre Anwendung beim Skalieren innerhalb der Instances, die die Benachrichtigung zur erneuten Verteilung empfangen, ordnungsgemäß herunterzufahren. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).

Im Folgenden finden Sie einige Gründe, warum Sie einen Lebenszyklus-Hook verwenden könnten:

- Für das ordnungsgemäße Herunterfahren von Amazon SQS-Workern
 - Zum Abschließen der Abmeldung vom Domain Name System (DNS)
 - Zum Abrufen und Hochladen von System- oder Anwendungsprotokollen in Amazon Simple Storage Service (Amazon S3)
3. Entwickeln Sie eine benutzerdefinierte Aktion für den Lebenszyklus-Hook. Um Ihre benutzerdefinierte Aktion so schnell wie möglich aufzurufen, müssen Sie wissen, wann eine Instance bereit ist, beendet zu werden. Sie finden dies heraus, indem Sie den Lebenszyklusstatus der Instance ermitteln.

- Um eine Aktion außerhalb der Instance aufzurufen, schreiben Sie eine EventBridge Regel und automatisieren Sie, welche Aktion ausgeführt werden soll, wenn ein Ereignismuster mit der Regel übereinstimmt.
- Um eine Aktion innerhalb der Instance aufzurufen, konfigurieren Sie die Instance so, dass sie ein Beendigungsskript ausführt und den Lebenszyklusstatus über die Instance-Metadaten abrufen.

Es ist wichtig, die benutzerdefinierte Aktion so zu gestalten, dass sie in weniger als zwei Minuten abgeschlossen ist. Dadurch wird sichergestellt, dass genügend Zeit zur Verfügung steht, um Aufgaben zu erledigen, bevor die Instance beendet wird.

Nachdem Sie diese Schritte abgeschlossen haben, können Sie mit den Kapazitätswiederherstellungen beginnen.

Verhalten bei Kapazitätswiederherstellungen

Bei Kapazitätswiederherstellungen verhält sich Amazon EC2 Auto Scaling folgendermaßen, wenn eine Instance eine Empfehlung zur erneuten Verteilung erhält:

- Beim Starten einer neuen Spot Instance wartet Amazon EC2 Auto Scaling, bis die neue Instance ihre Zustandsprüfung besteht, bevor die vorherige Instance beendet wird. Wenn mehr als eine Instance ersetzt wird, beginnt die Beendigung jeder vorherigen Instance, nachdem die neue Instance gestartet wurde und ihre Zustandsprüfung bestanden hat.
- Da Amazon EC2 Auto Scaling vor dem Beenden der vorherigen Instances versucht, neue zu starten, kann das Wiederherstellen des Gleichgewichts beeinträchtigt und sogar gänzlich unterbrochen werden, falls die angegebene maximale Kapazität nahezu oder gänzlich erreicht ist. Um dieses Problem zu vermeiden, kann Amazon EC2 Auto Scaling die maximale Größe der Gruppe vorübergehend bis zu 10 Prozent der gewünschten Kapazität überschreiten.
- Wenn Sie keinen Lebenszyklus-Hook zu Ihrer Auto-Scaling-Gruppe hinzufügen, beginnt Amazon EC2 Auto Scaling mit dem Beenden der vorherigen Instances, sobald die neuen Instances ihre Zustandsprüfung bestehen.
- Wenn Sie einen Lebenszyklus-Hook hinzugefügt haben, verlängert sich die Zeit, die benötigt wird, bis wir mit der Beendigung der vorherigen Instances beginnen, um den Timeout-Wert, den Sie für den Lebenszyklus-Hook angegeben haben.
- Wenn Sie Skalierungsrichtlinien oder eine geplante Skalierung verwenden, werden die Skalierungsaktivitäten parallel ausgeführt. Wenn eine Skalierungsaktivität ausgeführt wird und Ihre

Auto-Scaling-Gruppe unter der neuen gewünschten Kapazität liegt, wird Amazon EC2 Auto Scaling zuerst skaliert, bevor die vorherigen Instances beendet werden.

Wenn in einer Availability Zone keine Kapazität für Ihre Instance-Typen vorhanden ist, versucht Amazon EC2 Auto Scaling weiterhin, Spot-Instances in anderen aktivierten Availability Zones zu starten, bis es erfolgreich ist.

Im Worst-Case-Szenario, wenn die neuen Instances nicht gestartet werden oder die Zustandsprüfung fehlschlägt, versucht Amazon EC2 Auto Scaling weiterhin, sie neu zu starten. Während es versucht, neue Instances zu starten, werden Ihre vorherigen schließlich unterbrochen und mit einer zweiminütigen Unterbrechungsmeldung zwangsweise beendet.

Überlegungen

Berücksichtigen Sie bei der Verwendung von Kapazitätswiederherstellungen die folgenden Punkte:

Gestalten Sie Ihre Anwendung so, dass sie Spot-Unterbrechungen toleriert

Ihre Anwendung sollte dynamische Änderungen in der Anzahl der Instances und die Möglichkeit, dass eine Spot-Instance frühzeitig unterbrochen wird, bewältigen können. Wenn sich z. B. die Auto-Scaling-Gruppe hinter einem Elastic Load Balancing Load Balancer befindet, wartet Amazon EC2 Auto Scaling darauf, dass die Instance vom Load Balancer abgemeldet wird, bevor Ihr Lebenszyklus-Hook aufgerufen wird. Wenn die Zeit zum Abmelden der Instance und zum Abschließen der Lebenszyklus-Aktion zu lange dauert, wird die Instance möglicherweise unterbrochen, während Amazon EC2 Auto Scaling auf den Abschluss Ihrer Lebenszyklus-Aktion wartet, bevor es die Instance beendet.

Es ist Amazon EC2 nicht immer möglich, das Signal für die Neuausgleichsempfehlung vor der zweiminütigen Spot-Instance-Unterbrechungsbenachrichtigung zu senden. Daher kann das Empfehlungssignal für eine erneute Verteilung manchmal zusammen mit der zweiminütigen Unterbrechungsbenachrichtigung eingehen. In diesem Fall ruft Amazon EC2 Auto Scaling den Lebenszyklus-Hook auf und versucht, sofort eine neue Spot-Instance zu starten.

Vermeiden Sie ein erhöhtes Risiko einer Unterbrechung von Ersatz-Spot-Instances

Ihre Ersatz-Spot-Instances haben möglicherweise ein erhöhtes Risiko einer Unterbrechung, wenn Sie die `lowest-price`-Zuweisungsstrategie verwenden. Das liegt daran, dass wir Instances im preisgünstigsten Pool starten, der zu diesem Zeitpunkt über verfügbare Kapazität verfügt, auch wenn Ihre Ersatz-Spot-Instances wahrscheinlich kurz nach dem Start unterbrochen werden. Um

ein erhöhtes Unterbrechungsrisiko zu vermeiden, wird dringend empfohlen, die `lowest-price`-Zuweisungsstrategie nicht zu verwenden. Stattdessen empfehlen wir die `price-capacity-optimized`-Zuweisungsstrategie. Diese Strategie startet Ersatz-Spot-Instances in Spot-Pools, bei denen die Wahrscheinlichkeit einer Unterbrechung am geringsten ist und die den niedrigsten Preis haben. Daher ist es weniger wahrscheinlich, dass sie in naher Zukunft unterbrochen werden.

Amazon EC2 Auto Scaling startet eine neue Instance nur dann, wenn die Verfügbarkeit gleich oder besser ist

Eines der Ziele des Kapazitätsausgleichs ist die Verbesserung der Verfügbarkeit einer Spot Instance. Wenn eine vorhandene Spot Instance eine Neuausgleichsempfehlung erhält, startet Amazon EC2 Auto Scaling nur dann eine neue Instance, wenn die neue Instance dieselbe oder eine bessere Verfügbarkeit als die vorhandene Instance bietet. Wenn das Risiko einer Unterbrechung einer neuen Instance größer ist als das der vorhandenen Instance, startet Amazon EC2 Auto Scaling keine neue Instance. Amazon EC2 Auto Scaling wird die Spot-Kapazitätspools jedoch weiterhin auf der Grundlage der vom Amazon-EC2-Spot-Service bereitgestellten Informationen bewerten und eine neue Instance starten, falls sich die Verfügbarkeit verbessert.

Es besteht die Möglichkeit, dass Ihre vorhandene Instance unterbrochen wird, ohne dass Amazon EC2 Auto Scaling proaktiv eine neue Instance startet. In diesem Fall versucht Amazon EC2 Auto Scaling, eine neue Instance zu starten, sobald die Spot-Instance-Unterbrechungsmeldung eingeht. Das geschieht unabhängig davon, ob bei der neuen Instance ein hohes Unterbrechungsrisiko besteht.

Capacity Rebalancing erhöht nicht die Unterbrechungsrate Ihrer Spot-Instance

Wenn Sie Capacity Rebalancing aktivieren, wird Ihre [Spot-Instance-Unterbrechungsrate](#) (die Anzahl der Spot-Instances, die zurückgefordert werden, wenn Amazon EC2 die Kapazität zurück benötigt) nicht erhöht. Wenn der Kapazitätsausgleich jedoch feststellt, dass bei einer Instance das Risiko einer Unterbrechung besteht, versucht Amazon EC2 Auto Scaling sofort, eine neue Instance zu starten. Daher werden möglicherweise mehr Instances ersetzt, als wenn Sie darauf gewartet hätten, dass Amazon EC2 Auto Scaling eine neue Instance startet, nachdem die gefährdete Instance unterbrochen wurde.

Sie können zwar mehr Instances mit aktivierten Kapazitätswiederherstellungen ersetzen, profitieren jedoch davon, dass Sie eher proaktiv als reaktiv sind. Dadurch haben Sie mehr Zeit, Maßnahmen zu ergreifen, bevor Ihre Instances unterbrochen werden. Mit einer [Spot-Instance-Unterbrechungsbenachrichtigung](#) haben Sie normalerweise nur bis zu zwei Minuten Zeit, um Ihre Instance ordnungsgemäß herunterzufahren. Wenn die Kapazitätswiederherstellungen eine

neue Instance im Voraus starten, geben Sie vorhandenen Prozessen eine bessere Chance, auf Ihrer gefährdeten Instance abgeschlossen zu werden. Sie können auch mit dem Herunterfahren Ihrer Instance beginnen, verhindern, dass neue Arbeiten für Ihre gefährdete Instance geplant werden, und die neu gestartete Instance auf die Übernahme der Anwendung vorbereiten. Mit dem proaktiven Ersetzen durch Kapazitätswiederherstellungen profitieren Sie von einer reibungslosen Kontinuität.

Betrachten Sie als theoretisches Beispiel zur Demonstration der Risiken und Vorteile des Einsatzes von Kapazitätswiederherstellungen das folgende Szenario:

- 14:00 Uhr – Für Instance A wird eine Empfehlung zum erneuten Ausgleich empfangen, und Amazon EC2 Auto Scaling versucht sofort, eine Ersatz-Instance B zu starten, sodass Sie Zeit haben, Ihre Shutdown-Verfahren zu starten.
- 14:30 Uhr – Für Instance B wird eine Empfehlung zum erneuten Ausgleich empfangen, die durch Instance C ersetzt wird, sodass Sie Zeit haben, Ihre Shutdown-Verfahren zu starten.
- 14:32 Uhr — Wenn keine Kapazitätswiederherstellungen aktiviert wären und um 14:32 Uhr eine Spot-Instance-Unterbrechungsmeldung für Instance A eingegangen wäre, hätten Sie nur bis zu zwei Minuten Zeit gehabt, um Maßnahmen zu ergreifen. Instance A wäre jedoch bis zu diesem Zeitpunkt ausgeführt worden.

Aktivieren Sie den Kapazitätsausgleich, um gefährdete Spot-Instances proaktiv zu ersetzen

Sie können das AWS-Managementkonsole oder verwenden AWS CLI , um den Kapazitätsausgleich für Ihre Auto Scaling Scaling-Gruppe zu aktivieren. Wenn Capacity Rebalancing aktiviert ist, versucht Amazon EC2 Auto Scaling, proaktiv die Spot-Instances in Ihrer Gruppe zu ersetzen, die eine Empfehlung zur Neuverteilung von EC2-Instances erhalten haben.

Aktivieren des Kapazitätsausgleichs (Konsole)

Sie können den Kapazitätsausgleich aktivieren oder deaktivieren, wenn Sie eine Auto-Scaling-Gruppe erstellen oder aktualisieren.

So aktivieren Sie den Kapazitätsausgleich für eine neue Auto-Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.

3. Für Schritt 1: Startvorlage/-konfiguration auswählen, einen Namen für die Auto-Scaling-Gruppe eingeben, eine Startvorlage auswählen und dann die Option Weiter auswählen, um mit dem nächsten Schritt fortzufahren.
4. Für Schritt 2: Instance-Startoptionen auswählen, für die Anforderungen an den Instance-Typ Einstellungen auswählen, um eine gemischte Instance-Gruppe zu erstellen. Dazu gehören die Instance-Typen, die gestartet werden können, Instance-Kaufoptionen und Zuweisungsstrategien für Spot- und On-Demand-Instances. Standardmäßig sind diese Einstellungen nicht konfiguriert. Um sie zu konfigurieren, müssen Sie Override launch template (Startvorlage überschreiben) auswählen. Weitere Informationen zum Erstellen von Gruppen mit gemischten Instances finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).
5. Wählen Sie die gewünschten Optionen unter Netzwerk aus. Stellen Sie sicher, dass sich die Subnetze, die Sie verwenden möchten, in verschiedenen Availability Zones befinden.
6. Wählen Sie im Abschnitt Zuweisungsstrategien eine Spot-Zuweisungsstrategie aus. Um die Kapazitätswiederherstellungen zu aktivieren oder zu deaktivieren, müssen Sie das Kontrollkästchen unter Kapazitätswiederherstellungen aktivieren oder deaktivieren. Diese Option wird nur angezeigt, wenn Sie einen Prozentsatz der Auto-Scaling-Gruppe anfordern, der als Spot-Instances gestartet werden soll, im Abschnitt Instance-Kaufoptionen angeben.
7. Erstellen Sie die Auto-Scaling-Gruppe
8. (Optional) Fügen Sie nach Bedarf Lebenszyklus-Hooks hinzu. Weitere Informationen finden Sie unter [Fügen Sie Lifecycle-Hooks zu Ihrer Auto Scaling Scaling-Gruppe hinzu](#).

So aktivieren oder deaktivieren Sie den Kapazitätsausgleich für eine vorhandene Auto-Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe. Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.
3. Wählen Sie auf der Registerkarte Details die Optionen Allocation strategies (Zuweisungsstrategien), Edit (Bearbeiten) aus.
4. Aktivieren oder deaktivieren Sie im Abschnitt Zuweisungsstrategien die Kapazitätswiederherstellungen, indem Sie das Kontrollkästchen unter Kapazitätswiederherstellungen aktivieren oder deaktivieren.
5. Wählen Sie Aktualisieren aus.

Aktivieren Sie den Kapazitätsneuausgleich (AWS CLI)

Die folgenden Beispiele zeigen, wie Sie Capacity AWS CLI Rebalancing mithilfe von aktivieren und deaktivieren können.

Verwenden Sie den [update-auto-scaling-group](#) Befehl [create-auto-scaling-group](#) oder mit dem folgenden Parameter:

- `--capacity-rebalance/--no-capacity-rebalance`— Boolescher Wert, der angibt, ob Capacity Rebalancing aktiviert ist.

Bevor Sie den [create-auto-scaling-group](#) Befehl aufrufen, benötigen Sie den Namen einer Startvorlage, die für die Verwendung mit einer Auto Scaling Scaling-Gruppe konfiguriert ist. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).

Note

Das folgende Verfahren zeigt, wie Sie eine in JSON oder YAML formatierte Konfigurationsdatei verwenden. Wenn Sie AWS CLI Version 1 verwenden, müssen Sie eine Konfigurationsdatei im JSON-Format angeben. Wenn Sie AWS CLI Version 2 verwenden, können Sie eine Konfigurationsdatei angeben, die entweder in YAML oder JSON formatiert ist.

JSON

So erstellen und konfigurieren Sie eine neue Auto-Scaling-Gruppe

- Verwenden Sie den folgenden [create-auto-scaling-group](#) Befehl, um eine neue Auto Scaling Scaling-Gruppe zu erstellen und den Kapazitätsausgleich zu aktivieren. Durch diesen Befehl wird auf eine JSON-Datei als einziger Parameter für Ihre Auto-Scaling-Gruppe verwiesen.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

Wenn Sie noch nicht über eine CLI-Konfigurationsdatei verfügen, die eine [Richtlinie für gemischte Instances](#) angibt, erstellen Sie eine.

Fügen Sie die folgende Zeile zum übergeordneten JSON-Objekt in der Konfigurationsdatei hinzu.

```
{
  "CapacityRebalance": true
}
```

Im Folgenden sehen Sie ein Beispiel für eine `config.json`-Datei.

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredCapacity": 12,
  "MinSize": 12,
  "MaxSize": 15,
  "CapacityRebalance": true,
  "MixedInstancesPolicy": {
    "InstancesDistribution": {
      "OnDemandBaseCapacity": 0,
      "OnDemandPercentageAboveBaseCapacity": 25,
      "SpotAllocationStrategy": "price-capacity-optimized"
    },
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Default"
      },
      "Overrides": [
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        },
        {
          "InstanceType": "m5.large"
        },
        {
          "InstanceType": "m5a.large"
        },
        {
          "InstanceType": "c4.large"
        },
        {
          "InstanceType": "m4.large"
        }
      ]
    }
  }
}
```

```

        {
            "InstanceType": "c3.large"
        },
        {
            "InstanceType": "m3.large"
        }
    ]
},
"TargetGroupARNs": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff",
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```

YAML

So erstellen und konfigurieren Sie eine neue Auto-Scaling-Gruppe

- Verwenden Sie den folgenden [create-auto-scaling-group](#) Befehl, um eine neue Auto Scaling Scaling-Gruppe zu erstellen und den Kapazitätsausgleich zu aktivieren. Dieser Befehl verweist auf eine YAML-Datei als einziger Parameter für Ihre Auto-Scaling-Gruppe.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

Fügen Sie der in YAML formatierten Konfigurationsdatei die folgende Zeile hinzu.

```
CapacityRebalance: true
```

Im Folgenden sehen Sie ein Beispiel für eine config.yaml-Datei.

```

---
AutoScalingGroupName: my-asg
DesiredCapacity: 12
MinSize: 12
MaxSize: 15
CapacityRebalance: true
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 0
    OnDemandPercentageAboveBaseCapacity: 25

```

```

SpotAllocationStrategy: price-capacity-optimized
LaunchTemplate:
  LaunchTemplateSpecification:
    LaunchTemplateName: my-launch-template
    Version: $Default
  Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  TargetGroupARNs:
    - arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff
  VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782

```

So aktivieren Sie den Kapazitätsausgleich für eine vorhandene Auto-Scaling-Gruppe

- Verwenden Sie den folgenden [update-auto-scaling-group](#) Befehl, um Capacity Rebalancing zu aktivieren.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
--capacity-rebalance
```

So überprüfen Sie, ob der Kapazitätsausgleich für eine Auto-Scaling-Gruppe aktiviert ist

- Verwenden Sie den folgenden [describe-auto-scaling-groups](#) Befehl, um zu überprüfen, ob der Kapazitätsausgleich aktiviert ist, und um die Details anzuzeigen.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
```

```
        "AutoScalingGroupARN": "arn",
        ...
        "CapacityRebalance": true
    }
]
}
```

So deaktivieren Sie den Neuausgleich der Kapazität

Verwenden Sie den [update-auto-scaling-group](#) Befehl mit der `--no-capacity-rebalance` Option, um den Kapazitätsausgleich zu deaktivieren.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
--no-capacity-rebalance
```

Zugehörige Ressourcen

Weitere Informationen zum Kapazitätsausgleich finden Sie im Compute-Blog unter [Proaktive Verwaltung des Spot-Instance-Lebenszyklus mithilfe der neuen Capacity Rebalancing-Funktion für Amazon EC2 Auto Scaling](#). AWS

Weitere Informationen zu den Empfehlungen zur Neuverteilung von EC2-Instances finden Sie unter Empfehlungen zur [Neuverteilung von EC2-Instances](#) im Amazon EC2-Benutzerhandbuch.

Weitere Informationen zu Lebenszyklus-Hooks finden Sie in den folgenden Ressourcen.

- [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#)(verwenden) EventBridge
- [Tutorial: Verwenden Sie Datenskript- und Instance-Metadaten, um den Lebenszyklusstatus abzurufen](#)

Einschränkungen

- Amazon EC2 Auto Scaling kann die Instance, welche die Neuausgleichsbenachrichtigung erhält, nur beenden, wenn die Instance nicht vor dem Abskalieren geschützt ist. Der Abskalieren-Schutz verhindert jedoch nicht, dass eine Beendigung aufgrund einer Spot-Unterbrechung erfolgt. Weitere Informationen finden Sie unter [Verwenden Sie den Instance Scale-In Protection, um die Instanzbeendigung zu kontrollieren](#).

- Support für Kapazitätswiederherstellungen ist in allen kommerziellen AWS-Regionen verfügbar, in denen Amazon EC2 Auto Scaling verfügbar ist, mit Ausnahme der Region Naher Osten (VAE).

Reservieren Sie Kapazität in bestimmten Availability Zones mit Kapazitätsreservierungen

Mit Amazon EC2 On-Demand-Kapazitätsreservierungen können Sie Rechenkapazität in bestimmten Availability Zones reservieren. Um mit der Verwendung von Kapazitätsreservierungen mit Auto Scaling zu beginnen, erstellen Sie zunächst eine Kapazitätsreservierung oder eine Kapazitätsreservierungsgruppe in einer bestimmten Availability Zone. Anschließend können Sie Ihrer Auto Scaling Scaling-Gruppe eine Einstellung zur Kapazitätsreservierung hinzufügen, wenn Sie sie erstellen oder wenn Sie eine bestehende Gruppe aktualisieren.


Informationen zum Erstellen einer Kapazitätsreservierung finden [Sie unter Kapazitätsreservierung erstellen](#) im Amazon EC2 EC2-Benutzerhandbuch. Informationen zum Erstellen einer Kapazitätsreservierungsgruppe finden Sie unter [Erstellen einer Kapazitätsreservierungsgruppe](#) im Amazon EC2 EC2-Benutzerhandbuch.

Einstellung „Kapazitätsreservierung“

Die Einstellung „Kapazitätsreservierung“ hilft Ihnen bei der effizienten Nutzung von Kapazitätsreservierungen, indem Sie die reservierte Kapazität in einer Kapazitätsreservierung priorisieren, bevor Sie On-Demand-Kapazitäten verwenden. Sie können aus den folgenden Präferenzoptionen für die Kapazitätsreservierung wählen:

- Standard — Auto Scaling verwendet die Einstellung Kapazitätsreservierung aus Ihrer Startvorlage oder eine offene Kapazitätsreservierung.
- Keine — Auto Scaling startet keine Instances in einer Kapazitätsreservierung. Instances werden in On-Demand-Kapazität ausgeführt.
- Nur Kapazitätsreservierungen — Auto Scaling startet nur Instances in einer Kapazitätsreservierungs- oder Kapazitätsreservierungsgruppe. Wenn keine Kapazität verfügbar ist, können Instances nicht gestartet werden.
- Kapazitätsreservierungen zuerst — Auto Scaling startet Instances in einer Kapazitätsreservierungs- oder Kapazitätsreservierungsgruppe. Wenn keine Kapazität verfügbar ist, werden Instances als On-Demand-Kapazität ausgeführt.

Wenn Sie „Nur Kapazitätsreservierungen“ oder „Kapazitätsreservierungen zuerst“ auswählen, können Sie ein Kapazitätsreservierungsziel angeben.

 Note

Sie müssen eine Einstellung für die Kapazitätsreservierung auswählen. Das Kapazitätsreservierungsziel ist optional.

Überlegungen zur Präferenz für Kapazitätsreservierungen und zu Startvorlagen

Beachten Sie Folgendes, wenn Sie „Nur Kapazitätsreservierungen“ oder „Kapazitätsreservierungen zuerst“ auswählen:

- Wenn Sie „Nur Kapazitätsreservierungen“ oder „Kapazitätsreservierungen zuerst“ auswählen, verwendet Auto Scaling das in der Auto-Scaling-Gruppe angegebene Kapazitätsreservierungsziel anstelle des in der Startvorlage angegebenen Kapazitätsreservierungsziels.
- Wenn Sie „Nur Kapazitätsreservierungen“ oder „Kapazitätsreservierungen zuerst“ auswählen und kein Kapazitätsreservierungsziel angeben, verwendet Auto Scaling die Startvorlage „Kapazitätsreservierungsziel“ oder eine offene Kapazitätsreservierung.

Zielspezifikation für die Kapazitätsreservierung

Wenn Sie nur Kapazitätsreservierungen oder zuerst Kapazitätsreservierungen auswählen, sind die folgenden Zieloptionen für Kapazitätsreservierungen verfügbar:

- Offen — Auto Scaling startet Instances in jeder offenen Kapazitätsreservierung. Wenn Sie „Nur Kapazitätsreservierungen“ ausgewählt haben und keine Kapazität verfügbar ist, können Instances nicht gestartet werden. Wenn Sie zuerst Kapazitätsreservierungen ausgewählt haben und keine Kapazität verfügbar ist, werden Instances als On-Demand-Kapazität gestartet.
- Kapazitätsreservierung angeben — Auto Scaling startet Instances in der angegebenen Kapazitätsreservierung. Wenn Sie „Nur Kapazitätsreservierungen“ ausgewählt haben und keine Kapazität verfügbar ist, können Instances nicht gestartet werden. Wenn Sie zuerst Kapazitätsreservierungen ausgewählt haben und keine Kapazität verfügbar ist, werden Instances als On-Demand-Kapazität gestartet.
- Ressourcengruppe für Kapazitätsreservierung angeben — Auto Scaling startet Instances in einer offenen Kapazitätsreservierung in der angegebenen Kapazitätsreservierungsressourcengruppe.

Wenn Sie „Nur Kapazitätsreservierungen“ ausgewählt haben und keine Kapazität verfügbar ist, können Instances nicht gestartet werden. Wenn Sie zuerst Kapazitätsreservierungen ausgewählt haben und keine Kapazität verfügbar ist, werden Instances als On-Demand-Kapazität gestartet.

Balance zwischen Verfügbarkeitszonen und Kapazitätsreservierungen

Auto Scaling priorisiert das Gleichgewicht in der Availability Zone, auch wenn Sie die Einstellungen für Kapazitätsreservierungen verwenden. Das bedeutet Folgendes:

- Mit `capacity-reservations-first` — Die Auto Scaling Scaling-Gruppe verteilt die Instances AZs zunächst gleichmäßig auf alle und verwendet dann, sofern verfügbar, Kapazitätsreservierungen. Wenn Kapazitätsreservierungen in einer AZ nicht verfügbar sind, werden Instances als On-Demand-Variante gestartet, um das AZ-Guthaben aufrechtzuerhalten.
- Mit `capacity-reservations-only` — Die Auto Scaling Scaling-Gruppe verwendet nur Kapazitätsreservierungen. Dies kann zu einer ungleichmäßigen Verteilung der Instanzen führen, AZs je nachdem, wo Kapazitätsreservierungen verfügbar sind.

Beispiel

Wenn Sie 10 Kapazitätsreservierungen in Az-a, 3 in Az-B, 1 in Az-C haben und eine gewünschte Kapazität von 9 Instances haben:

- Eine Nutzung von `capacity-reservations-first` 9 Instances führt zu 3 Instances pro AZ (Aufrechterhaltung des AZ-Guthabens), wobei einige Instances möglicherweise als On-Demand-Instances ausgeführt werden.
- Die Nutzung `capacity-reservations-only` für 9 Instances führt zu einer ungleichmäßigen Verteilung auf der Grundlage der verfügbaren Kapazitätsreservierungen.

Verwenden Sie die Präferenz „Kapazitätsreservierung“ mit Ihrer Auto Scaling Scaling-Gruppe

Um Kapazitätsreservierungen mit Ihrer Auto Scaling Scaling-Gruppe zu verwenden, müssen Sie zunächst eine Ressourcengruppe „Kapazitätsreservierung“ oder „Kapazitätsreservierung“ erstellen. Anschließend können Sie Ihrer Auto Scaling Scaling-Gruppe eine Einstellung zur Kapazitätsreservierung hinzufügen, wenn Sie sie erstellen oder wenn Sie eine bestehende Gruppe aktualisieren.

Informationen zum Erstellen einer Kapazitätsreservierung finden Sie unter [Kapazitätsreservierung erstellen](#) im Amazon EC2 EC2-Benutzerhandbuch. Informationen zum Erstellen einer Kapazitätsreservierungsgruppe finden Sie unter [Erstellen einer Kapazitätsreservierungsgruppe](#) im Amazon EC2 EC2-Benutzerhandbuch.

Verwenden Sie eine der folgenden Methoden, um die Einstellung „Kapazitätsreservierung“ zu verwenden, wenn Sie eine Auto Scaling Scaling-Gruppe erstellen oder bearbeiten.

Console

Um die Einstellung „Kapazitätsreservierung“ in einer neuen Gruppe (Konsole) zu verwenden

1. Folgen Sie den Anweisungen [Erstellen einer Auto-Scaling-Gruppe mithilfe des Amazon EC2-Startassistenten](#) unter und schließen Sie jeden Schritt des Verfahrens bis zu Schritt 3 ab.
2. Wählen Sie auf der Seite Gruppengröße und Skalierung konfigurieren unter Zusätzliche Kapazitätseinstellungen, Einstellung Kapazitätsreservierung eine Einstellung für die Kapazitätsreservierung aus. Weitere Informationen zur Präferenz „Kapazitätsreservierung“ finden Sie unter [Einstellung „Kapazitätsreservierung“](#).
3. Fahren Sie mit den Schritten unter [Erstellen einer Auto-Scaling-Gruppe mithilfe des Amazon EC2-Startassistenten](#) fort.

AWS CLI

So verwenden Sie die Einstellung „Kapazitätsreservierung“ für eine neue Gruppe (AWS CLI)

Fügen Sie dem [create-auto-scaling-group](#)-Befehl den `--capacity-reservation-specification`-Parameter hinzu.

1. Geben Sie eine Präferenz für die Kapazitätsreservierung an. Weitere Informationen finden Sie unter [Einstellung „Kapazitätsreservierung“](#).
2. Geben Sie ein Ziel für die Kapazitätsreservierung an. Wenn Sie „Nur Kapazitätsreservierungen“ oder „Kapazitätsreservierungen zuerst“ auswählen und kein Kapazitätsreservierungsziel angeben, verwendet Auto Scaling die Startvorlage „Kapazitätsreservierungsziel“ oder eine offene Kapazitätsreservierung.

Console

Um die Einstellung „Kapazitätsreservierung“ für eine bestehende Gruppe (Konsole) zu verwenden

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die AWS-Region aus, in der Sie Ihre Auto-Scaling-Gruppe erstellt haben.
3. Aktivieren Sie das Kontrollkästchen neben der Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

4. Wählen Sie auf der Registerkarte Details unter Einstellung Kapazitätsreservierung die Option Bearbeiten aus.
5. Wählen Sie unter Zusätzliche Kapazitätseinstellungen, Präferenz für Kapazitätsreservierung eine Einstellung für Kapazitätsreservierung aus. Weitere Informationen zur Präferenz „Kapazitätsreservierung“ finden Sie unter [Einstellung „Kapazitätsreservierung“](#).
6. Wählen Sie Aktualisieren aus.

AWS CLI

So verwenden Sie die Einstellung „Kapazitätsreservierung“ für eine bestehende Gruppe (AWS CLI)

Fügen Sie dem [update-auto-scaling-group](#)-Befehl den `--capacity-reservation-specification`-Parameter hinzu.

1. Geben Sie eine Präferenz für die Kapazitätsreservierung an. Weitere Informationen finden Sie unter [Einstellung „Kapazitätsreservierung“](#).
2. Geben Sie ein Ziel für die Kapazitätsreservierung an. Wenn Sie „Nur Kapazitätsreservierungen“ oder „Kapazitätsreservierungen zuerst“ auswählen und kein Kapazitätsreservierungsziel angeben, verwendet Auto Scaling die Startvorlage „Kapazitätsreservierungsziel“ oder eine offene Kapazitätsreservierung.

Unterbrechungsfreie Kapazitätsreservierungen mit EC2 Auto Scaling

Bei unterbrechbaren Kapazitätsreservierungen handelt es sich um Kapazitätsreserven, die Ihnen von Eigentümern für On-Demand-Kapazitätsreservierungen innerhalb Ihres Unternehmens zur

Verfügung gestellt wurden. AWS Diese Reservierungen eignen sich für unterbrechbare Workloads, da der Kapazitätsinhaber die Kapazität jederzeit zurückfordern kann, indem er eine zweiminütige Unterbrechungsbenachrichtigung einreicht. Danach werden die Instances beendet. Einen Überblick über Reservierungen für unterbrechbare Kapazitäten und deren Funktionsweise finden Sie unter Reservierungen für [unterbrechbare Kapazitäten](#) im Amazon EC2 EC2-Benutzerhandbuch.

Zunächst erstellen Sie eine unterbrechbare Reservierung, indem Sie ungenutzte Kapazität aus einer beliebigen On-Demand-Kapazitätsreservierung in Ihrem Konto zuweisen. Als Nächstes erstellen Sie Ihre Startvorlage, geben die Reservierungs-ID für unterbrechbare Kapazitäten und den entsprechenden Markttyp an. Erstellen Sie abschließend Ihre Auto Scaling Scaling-Gruppe, um die von Ihnen erstellte Startvorlage zu verwenden.

Note

- Gruppen mit gemischten Instanzen werden bei unterbrechbaren Kapazitätsreservierungen nicht unterstützt.
- Die Reservierungen für unterbrechbare Kapazitäten werden in Ihrem Konto mit einem neuen Attribut angezeigt, das auf `interruptible` gesetzt ist. `true`

Themen

- [Betriebliche Richtlinien](#)
- [Geben Sie in Ihrer Startvorlage eine Reservierung für unterbrechbare Kapazität an](#)
- [Einschränkungen](#)
- [Zugehörige Ressourcen](#)

Betriebliche Richtlinien

Im Folgenden finden Sie grundlegende Betriebsrichtlinien, die Sie beachten sollten, wenn Sie Unterbrechungskapazitätsreservierungen mit einer Auto Scaling Scaling-Gruppe verwenden.

- Instances, die aufgrund von Reservierungen für unterbrechbare Kapazität gestartet wurden, können mit einer Frist von 2 Minuten gekündigt werden, wenn die Kapazität zurückgewonnen wird. Entwerfen Sie Ihre Anwendungen so, dass sie problemlos heruntergefahren werden können.

- Amazon EC2 sendet EventBridge Benachrichtigungen 2 Minuten vor dem Beenden von Instances. Auto Scaling reagiert zwar nicht automatisch auf diese Ereignisse, Sie können jedoch eine benutzerdefinierte Automatisierung einrichten, um auf Unterbrechungsmeldungen zu reagieren.
- Wenn Instances aufgrund einer Kapazitätsrückgewinnung beendet werden, erkennt der Health Checker von Auto Scaling die Kündigung und startet automatisch Ersatz-Instances entsprechend der Konfiguration Ihrer Gruppe.
- Um die Konsistenz zu gewährleisten, verweisen Sie auf eine bestimmte Version der Startvorlage statt auf die `$Latest` Version `$Default` oder.

Note

Wenn eine Reservierungsinstanz für unterbrechbare Kapazität zurückerobert wird, wird in den Skalierungsaktivitäten für Ihre Auto Scaling Scaling-Gruppe Folgendes angegeben: `taken out of service in response to an Amazon EC2 health check that indicated it had been terminated or stopped` Auto Scaling versucht, die Instance auf dieselbe Weise zu ersetzen, wie es bei jeder Instance der Fall ist, die eine Zustandsprüfung nicht besteht.

Geben Sie in Ihrer Startvorlage eine Reservierung für unterbrechbare Kapazität an

Verwenden Sie eine der folgenden Methoden, um eine Startvorlage zu erstellen, die auf eine bestimmte unterbrechbare Kapazitätsreservierung für Ihre Auto Scaling Scaling-Gruppe abzielt:

Console

Um eine Reservierung für unterbrechbare Kapazitäten in Ihrer Startvorlage (Konsole) anzugeben

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie in der oberen Navigationsleiste den Ort aus, AWS-Region an dem Sie Ihre unterbrechbare Kapazität reserviert haben.
3. Wählen Sie im Navigationsbereich unter Instances die Option Launch Templates aus.
4. Wählen Sie Startvorlage erstellen und erstellen Sie die Startvorlage. Geben Sie bei Bedarf die ID des AMI, den Instance-Typ und alle anderen Einstellungen für die Startvorlage an.
5. Erweitern Sie den Abschnitt Erweiterte Details, um die erweiterten Einstellungen anzuzeigen.
6. Wählen Sie als Kaufoption die Option Unterbrechungsfreie Kapazitätsreservierung aus.

- Wählen Sie für Kapazitätsreservierung die Option Ziel nach ID und dann für Kapazitätsreservierung — Ziel nach ID die Kapazitätsreservierungs-ID einer bestehenden Reservierung für unterbrechbare Kapazität aus.
- Klicken Sie danach auf Startvorlage erstellen.

Hilfe zum Erstellen einer Auto Scaling Scaling-Gruppe mit einer Startvorlage finden Sie unter [Erstellen einer Auto-Scaling-Gruppe mithilfe einer Startvorlage](#).

AWS CLI

So geben Sie in Ihrer Startvorlage eine Reservierung für unterbrechbare Kapazitäten an ()AWS CLI

Verwenden Sie den folgenden [create-launch-template](#) Befehl, um eine Startvorlage zu erstellen, die eine bestehende Reservierungs-ID für unterbrechbare Kapazitäten angibt. Ersetzen Sie jedes Platzhalter für Benutzereingaben durch Ihre eigenen Informationen.

```
aws ec2 create-launch-template --launch-template-name my-template-for-interruptible-cr \  
--version-description AutoScalingVersion1 --region us-east-2 \  
--launch-template-data file://config.json
```

Tip

Wenn dieser Befehl einen Fehler auslöst, stellen Sie sicher, dass Sie den Befehl AWS CLI lokal auf die neueste Version aktualisiert haben.

Inhalt von `config.json`:

```
{  
  "ImageId": "ami-04d5cc9b88example",  
  "InstanceType": "t3.nano",  
  "SecurityGroupIds": [  
    "sg-903004f88example"  
  ],  
  "KeyName": "MyKeyPair",  
  "InstanceMarketOptions": {  
    "MarketType": "interruptible-capacity-reservation"  
  }  
}
```

```

    },
    "CapacityReservationSpecification": {
      "CapacityReservationTarget": {
        "CapacityReservationId": "cr-02168da1478b509e0"
      }
    }
  }
}

```

Das Folgende ist eine Beispielausgabe:

```

{
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-068f72b724example",
    "LaunchTemplateName": "my-template-for-interruptible-cr",
    "CreateTime": "2023-10-27T15:12:44.000Z",
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "DefaultVersionNumber": 1,
    "LatestVersionNumber": 1
  }
}

```

Sie können den folgenden [describe-launch-template-versions](#) Befehl verwenden, um die Reservierungs-ID für unterbrechbare Kapazitäten zu überprüfen, die der Startvorlage zugeordnet ist:

```

aws ec2 describe-launch-template-versions --launch-template-names my-template-for-interruptible-cr \
--region us-east-2

```

Auto Scaling Scaling-Konfiguration auf Gruppenebene

Sie können optional Einstellungen für die Kapazitätsreservierung auf Auto Scaling Scaling-Gruppenebene angeben, um die Einstellungen der Startvorlage zu überschreiben:

- **Bevorzugte Kapazitätsreservierung:** `capacity-reservations-only` Wird nur für unterbrechbare Kapazitätsreservierungen unterstützt. Die `capacity-reservations-first` Präferenz wird nicht unterstützt.
- **Zielspezifikation:** Sie können auf Gruppenebene eine andere Reservierungs-ID für unterbrechbare Kapazitäten angeben, um die Einstellung für die Startvorlage zu überschreiben.

Einschränkungen

- Richtlinien für gemischte Instanzen werden bei Reservierungen für unterbrechbare Kapazitäten nicht unterstützt.
- Sie müssen in Ihrer Startvorlage den richtigen Markttyp (`interruptible-capacity-reservation`) zusammen mit der Kapazitätsreservierungs-ID angeben.

Zugehörige Ressourcen

- Informationen zum Erstellen und Verwalten von Reservierungen für unterbrechbare Kapazitäten finden Sie unter Reservierungen für [unterbrechbare Kapazitäten](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Allgemeine Informationen zur Verwendung von Kapazitätsreservierungen mit Auto Scaling finden Sie unter [Reservieren Sie Kapazität in bestimmten Availability Zones mit Kapazitätsreservierungen](#).
- Informationen zu Integritätsprüfungen für Auto Scaling Scaling-Gruppen finden Sie unter [Integritätsprüfungen für Instances in einer Auto Scaling Scaling-Gruppe](#).

Erstellen Sie Auto Scaling Scaling-Gruppen über die Befehlszeile mit AWS CloudShell

Unter [unterstützt](#) können Sie AWS CLI Befehle ausführen AWS-Regionen, indem AWS CloudShell Sie eine browserbasierte, vorauthentifizierte Shell verwenden, die direkt von der aus gestartet wird. AWS-Managementkonsole Sie können AWS CLI Befehle für Dienste ausführen, indem Sie Ihre bevorzugte Shell (Bash- oder Z-Shell) verwenden. PowerShell

Sie können AWS CloudShell von der aus AWS-Managementkonsole mit einer der folgenden beiden Methoden starten:

- Wählen Sie das AWS CloudShell Symbol in der Navigationsleiste der Konsole. Es befindet sich rechts neben dem Suchfeld.
- Verwenden Sie das Suchfeld in der Navigationsleiste der Konsole, um nach der CloudShellOption zu suchen CloudShellund diese dann auszuwählen.

Beim ersten AWS CloudShell Start in einem neuen Browserfenster wird ein Begrüßungsfenster mit einer Liste der wichtigsten Funktionen angezeigt. Nachdem Sie dieses Panel geschlossen haben,

werden Statusaktualisierungen bereitgestellt, während die Shell Ihre Konsolenanmeldeinformationen konfiguriert und weiterleitet. Wenn die Eingabeaufforderung angezeigt wird, ist die Shell für die Interaktion bereit.

Weitere Informationen zu diesem Service finden Sie im [AWS CloudShell -Benutzerhandbuch](#).

Erstellen Sie Auto Scaling Scaling-Gruppen mit AWS CloudFormation

Amazon EC2 Auto Scaling ist in einen Service integriert AWS CloudFormation, der Sie bei der Modellierung und Einrichtung Ihrer AWS Ressourcen unterstützt, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die alle gewünschten AWS Ressourcen beschreibt (z. B. Auto Scaling Scaling-Gruppen) und diese Ressourcen für Sie CloudFormation bereitstellt und konfiguriert.

Wenn Sie es verwenden CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre Amazon EC2 Auto Scaling Scaling-Ressourcen konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Ressourcen einmal und stellen Sie dann dieselben Ressourcen immer wieder in mehreren Regionen AWS-Konten bereit.

Amazon EC2 Auto Scaling und Vorlagen CloudFormation

Um Ressourcen für Amazon EC2 Auto Scaling und damit verbundene Dienste bereitzustellen und zu konfigurieren, müssen Sie [CloudFormation -Vorlagen](#) verstehen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren CloudFormation Stacks bereitstellen möchten. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie CloudFormation Designer verwenden, um Ihnen die ersten Schritte mit Vorlagen zu erleichtern. CloudFormation Weitere Informationen finden Sie unter [Was ist CloudFormation Designer?](#) im AWS CloudFormation Benutzerhandbuch.

Bevor Sie mit der Erstellung Ihrer eigenen Stack-Vorlagen für Amazon EC2 Auto Scaling beginnen, führen Sie die folgenden Aufgaben durch:

- Erstellen Sie eine Startvorlage mit [AWS::EC2::LaunchTemplate](#).
- Erstellen Sie mithilfe von Group Group eine Auto Scaling [AWS::AutoScaling::AutoScaling](#).

Eine exemplarische Vorgehensweise, die Ihnen zeigt, wie Sie eine Auto-Scaling-Gruppe hinter einem Application Load Balancer bereitstellen, finden Sie unter [Exemplarische Vorgehensweise: Erstellen einer skalierten Anwendung mit Lastenausgleich](#) im AWS CloudFormation -Benutzerhandbuch.

Weitere nützliche Beispiele für Vorlagenausschnitte, mit denen Auto Scaling Scaling-Gruppen erstellt werden, und verwandte Ressourcen finden Sie in den folgenden Abschnitten des AWS CloudFormation Benutzerhandbuchs:

- Referenz zum [Amazon EC2 Auto Scaling Scaling-Ressourcentyp Referenz](#) zum
- [Konfigurieren Sie Amazon EC2 Auto Scaling Scaling-Ressourcen mit AWS CloudFormation](#)

Erfahren Sie mehr über CloudFormation

Weitere Informationen CloudFormation finden Sie in den folgenden Ressourcen:

- [AWS CloudFormation](#)
- [AWS CloudFormation Benutzerhandbuch](#)
- [CloudFormation API Reference](#)
- [AWS CloudFormation Benutzerhandbuch für die Befehlszeilenschnittstelle](#)

Holen Sie sich Empfehlungen zum Instanztyp mit AWS Compute Optimizer

AWS bietet Empfehlungen für den Amazon EC2 EC2-Instance-Typ, um Ihnen zu helfen, die Leistung zu verbessern, Geld zu sparen oder beides zu tun, indem Sie Funktionen verwenden, die von bereitgestellt werden AWS Compute Optimizer. Anhand dieser Empfehlungen können Sie entscheiden, ob Sie zu einem neuen Instance-Typ in Ihrer Auto Scaling Scaling-Gruppe wechseln möchten.

Um Empfehlungen abzugeben, analysiert Compute Optimizer Ihre vorhandenen Instance-Spezifikationen und den letzten Metrikverlauf. Die kompilierten Daten werden dann verwendet, um zu empfehlen, welche Amazon-EC2-Instance-Typen am besten für die Verarbeitung der vorhandenen Leistungs-Workload optimiert sind. Empfehlungen werden zusammen mit den Preisen der Instance pro Stunde zurückgegeben.

Note

Um Empfehlungen von Compute Optimizer zu erhalten, müssen Sie sich zunächst bei Compute Optimizer anmelden. Weitere Informationen finden Sie unter [Erste Schritte in AWS Compute Optimizer](#) im AWS Compute Optimizer -Benutzerhandbuch.

Inhalt

- [Einschränkungen](#)
- [Ergebnisse](#)
- [Anzeigen von Empfehlungen](#)
- [Überlegungen zur Bewertung der Empfehlungen](#)

Einschränkungen

Compute Optimizer generiert Empfehlungen für Instances in Auto-Scaling-Gruppen, die zum Starten und Ausführen von M-, C-, R-, T- und X-Instance-Typen konfiguriert sind. Es werden jedoch keine Empfehlungen für -g-Instance-Typen generiert, die auf AWS Graviton2-Prozessoren (z. B. C6g) basieren, und für -n-Instance-Typen, die eine höhere Netzwerkbandbreitenleistung aufweisen (z. B. M5n).

Die Auto-Scaling-Gruppen müssen auch so konfiguriert sein, dass sie einen einzelnen Instance-Typ ausführen (d. h. keine gemischten Instance-Typen), dürfen keiner Skalierungsrichtlinie zugeordnet sein und müssen dieselben Werte für die gewünschte, minimale und maximale Kapazität aufweisen (d. h. eine Auto-Scaling-Gruppe mit einer festen Anzahl von Instances). Compute Optimizer generiert Empfehlungen für Instances in Auto-Scaling-Gruppen, die alle dieser Konfigurationsanforderungen erfüllen.

Ergebnisse

Compute Optimizer klassifiziert die Ergebnisse für Auto-Scaling-Gruppen wie folgt:

- Nicht optimiert – Eine Auto-Scaling-Gruppe gilt als nicht optimiert, wenn Compute Optimizer eine Empfehlung identifiziert hat, die eine bessere Leistung für Ihr Workload bieten kann.
- Optimiert – Eine Auto-Scaling-Gruppe wird als optimiert angesehen, wenn Compute Optimizer feststellt, dass die Gruppe korrekt bereitgestellt ist, um Ihr Workload auszuführen, basierend auf

dem gewählten Instance-Typ. Für optimierte Ressourcen empfiehlt Compute Optimizer manchmal einen Instance-Typ der neuen Generation.

- Keine – Für diese Auto-Scaling-Gruppe liegen keine Empfehlungen vor. Dies kann vorkommen, wenn Sie bei Computer Optimizer weniger als 12 Stunden angemeldet waren oder die Auto-Scaling-Gruppe weniger als 30 Stunden ausgeführt wurde oder wenn die Auto-Scaling-Gruppe oder der Instance-Typ von Compute Optimizer nicht unterstützt wird. Weitere Informationen finden Sie im Abschnitt [Einschränkungen](#).

Anzeigen von Empfehlungen

Nachdem Sie sich für Compute Optimizer entschieden haben, können Sie die Ergebnisse und Empfehlungen anzeigen, die für Ihre Auto-Scaling-Gruppen generiert werden. Wenn Sie sich kürzlich angemeldet haben, werden Empfehlungen möglicherweise bis zu 12 Stunden nicht angezeigt.

So zeigen Sie Empfehlungen an, die für eine Auto-Scaling-Gruppe generiert wurden

1. Öffnen Sie die Compute Optimizer Optimizer-Konsole unter <https://console.aws.amazon.com/compute-optimizer/>.

Die Dashboard-Seite wird geöffnet.

2. Wählen Sie View recommendations for all Auto Scaling groups (Empfehlungen für alle Auto-Scaling-Gruppen anzeigen) aus.
3. Wählen Sie Ihre Auto-Scaling-Gruppe aus.
4. Wählen Sie die Option View details (Details anzeigen) aus.

Die Ansicht ändert sich, um bis zu drei verschiedene Instance-Empfehlungen in einer vorkonfigurierten Ansicht anzuzeigen, basierend auf den Standard-Tabelleneinstellungen. Es stellt auch aktuelle CloudWatch Metrikdaten (durchschnittliche CPU-Auslastung, durchschnittlicher Netzwerkeingang und durchschnittlicher Netzwerkausgang) für die Auto Scaling Scaling-Gruppe bereit.

Legen Sie fest, ob Sie eine der Empfehlungen verwenden möchten. Entscheiden Sie, ob Sie mit Blick auf eine Leistungssteigerung, eine Kostensenkung oder beides optimieren möchten.

Um den Instance-Typ in Ihrer Auto-Scaling-Gruppe zu ändern, aktualisieren Sie die Startvorlage oder aktualisieren Sie die Auto-Scaling-Gruppe so, dass sie eine neue Startkonfiguration verwendet. Für bestehende Instances wird weiterhin die vorherige Konfiguration verwendet. Um die vorhandenen

Instances zu aktualisieren, beenden Sie sie, damit sie durch Ihre Auto-Scaling-Gruppe ersetzt werden. Sie können auch zulassen, dass die Auto-Scaling-Gruppe ältere Instances schrittweise durch neuere Instances basierend auf Ihren [Beendigungsrichtlinien](#) ersetzt.

Note

Mit den Funktionen für maximale Instance-Lebensdauer und Instance-Aktualisierung können Sie auch vorhandene Instances in Ihrer Auto-Scaling-Gruppe ersetzen, um neue Instances zu starten, die die neue Startvorlage oder Startkonfiguration verwenden. Weitere Informationen erhalten Sie unter [Auto-Scaling-Instances basierend auf der maximalen Instance-Lebensdauer ersetzen](#) und [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).

Überlegungen zur Bewertung der Empfehlungen

Bevor Sie zu einem neuen Instance-Typ wechseln, sollten Sie Folgendes beachten:

- Die Empfehlungen prognostizieren nicht Ihre Nutzung. Die Empfehlungen basieren auf Ihrer bisherigen Nutzung während des letzten 14-Tage-Zeitraums. Stellen Sie sicher, dass Sie einen Instance-Typ auswählen, der Ihren zukünftigen Verwendungsanforderungen entspricht.
- Konzentrieren Sie sich auf die grafisch dargestellten Metriken, um zu ermitteln, ob die tatsächliche Nutzung geringer als die Instance-Kapazität ist. Sie können auch Metrikdaten (Durchschnitt, Spitze, Perzentil) einsehen, CloudWatch um Ihre EC2-Instance-Empfehlungen weiter auszuwerten. Beachten Sie zum Beispiel, wie sich die prozentualen CPU-Prozentsatzmetriken im Laufe des Tages verändern und ob es Datenverkehrsspitzen gibt, die berücksichtigt werden müssen. Weitere Informationen finden Sie unter [Verfügbare Messwerte anzeigen](#) im CloudWatch Amazon-Benutzerhandbuch.
- Compute Optimizer bietet möglicherweise Empfehlungen für Instances mit Spitzenlastleistung, bei denen es sich um T3-, T3a- und T2-Instances handelt. Wenn Sie Ihren Ausgangswert regelmäßig überschreiten, stellen Sie sicher, dass Sie dies auch weiterhin tun können, basierend auf der Zahl v CPUs des neuen Instance-Typs. Weitere Informationen finden Sie unter [CPU-Guthaben und Basisleistung für Burstable-Performance-Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Wenn Sie eine Reserved Instance erworben haben, wird Ihnen Ihre On-Demand-Instance möglicherweise als Reserved Instance in Rechnung gestellt. Bevor Sie Ihren aktuellen Instance-Typ ändern, sollten Sie zunächst die Auswirkungen auf die Nutzung und Abdeckung der Reserved Instance bewerten.

- Ziehen Sie nach Möglichkeit einen Umstieg auf Instances der neueren Generation in Betracht.
- Bei der Migration auf eine andere Instance-Familie ist darauf zu achten, dass der aktuelle Instance-Typ und der neue Instance-Typ miteinander kompatibel sind, z. B. in Bezug auf Virtualisierung, Architektur oder Netzwerktyp. Weitere Informationen finden Sie unter [Kompatibilität bei der Größenänderung von Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Berücksichtigen Sie abschließend die Bewertung des Leistungsrisikos, die für jede Empfehlung angegeben ist. Das Leistungsrisiko gibt den Aufwand an, den Sie möglicherweise aufwenden müssen, um zu überprüfen, ob der empfohlene Instance-Typ den Leistungsanforderungen Ihrem Workload entspricht. Darüber hinaus empfehlen wir, vor und nach jeder Änderung Last- und Leistungstests durchzuführen.

Weitere Ressourcen

Weitere Informationen zu den Themen auf dieser Seite finden Sie in den folgenden Quellen:

- [Amazon-EC2-Instance-Typen](#)
- [AWS Compute Optimizer Benutzerhandbuch](#)

Verwenden Sie Elastic Load Balancing, um den eingehenden Anwendungsdatenverkehr in Ihrer Auto Scaling Scaling-Gruppe zu verteilen

Mit Elastic Load Balancer können Sie den eingehenden Datenverkehr der Anwendung automatisch auf sämtliche EC2-Instances verteilen, die Sie ausführen. Sie können mit Elastic Load Balancing eingehende Anforderungen verwalten, indem Sie den Datenverkehr optimal weiterleiten, sodass keine Instance überfordert ist. Um Elastic Load Balancing mit Ihrer Auto-Scaling-Gruppe zu verwenden, [fügen Sie den Load Balancer an Ihre Auto-Scaling-Gruppe an](#). Damit wird die Auto-Scaling-Gruppe beim Load Balancer registriert, der als einziger Kontaktpunkt für den gesamten eingehenden Datenverkehr zu den Instances in Ihrer Auto-Scaling-Gruppe fungiert.

Wenn Sie Ihren Elastic Load Balancer mit einer Auto-Scaling-Gruppe verwenden, ist es nicht erforderlich, die EC2-Instances beim Load Balancer oder der Zielgruppe anzumelden. Instances, die von Ihrer Auto-Scaling-Gruppe gestartet werden, werden automatisch beim Load Balancer registriert. Ebenso werden Instances, die durch Ihre Auto-Scaling-Gruppe beendet werden, automatisch vom Load Balancer abgemeldet.

Nachdem Sie einen Load Balancer an Ihre Auto-Scaling-Gruppe angefügt haben, können Sie Ihre Auto-Scaling-Gruppe so konfigurieren, dass Elastic Load Balancing-Metriken (wie die Application Load Balancer-Anforderungsanzahl pro Ziel) verwendet werden, um die Anzahl der Instances in der Gruppe zu skalieren, wenn der Bedarf schwankt.

Optional können Sie Elastic Load Balancing-Zustandsprüfungen zu Ihrer Auto-Scaling-Gruppe hinzufügen, damit Amazon EC2 Auto Scaling auf der Grundlage dieser zusätzlichen Zustandsprüfungen fehlerhafte Instances identifizieren und ersetzen kann. Andernfalls können Sie einen CloudWatch Alarm einrichten, der Sie benachrichtigt, wenn die Anzahl gesunder Hosts in der Zielgruppe niedriger als zulässig ist.

Inhalt

- [Arten von Elastic Load Balancing](#)
- [Bereiten Sie das Anhängen eines Elastic Load Balancing Balancing-Load Balancers vor](#)
- [Fügen Sie Ihrer Auto Scaling Scaling-Gruppe einen Elastic Load Balancing Load Balancer hinzu](#)
- [Konfigurieren Sie einen Application Load Balancer oder Network Load Balancer von der Konsole aus](#)
- [Überprüfen des Anhangsstatus Ihres Load Balancers](#)
- [Fügen Sie eine Availability Zone hinzu](#)
- [Entfernen einer Availability Zone](#)
- [Trennen Sie eine Zielgruppe oder einen Classic Load Balancer von Ihrer Auto Scaling Scaling-Gruppe](#)
- [Beispiele für die Arbeit mit Elastic Load Balancing unter Verwendung der AWS CLI](#)

Arten von Elastic Load Balancing

Elastic Load Balancing bietet vier Arten von Load Balancern, die mit Ihrer Auto-Scaling-Gruppe verwendet werden können: Application Load Balancer, Network Load Balancer, Gateway Load Balancer und Classic Load Balancer.

Es gibt einen entscheidenden Unterschied in der Konfiguration der Load Balancer-Typen. Bei Application Load Balancern, Network Load Balancern und Gateway Load Balancern werden Instances als Ziele bei einer Zielgruppe registriert, und Sie leiten den Datenverkehr an die Zielgruppe weiter. Bei Classic Load Balancern werden Instances direkt beim Load Balancer registriert.

Application Load Balancer

Führt das Routing und den Lastenausgleich auf Anwendungsebene (HTTP/HTTPS) durch und unterstützt das pfadbasierte Routing. Ein Application Load Balancer kann Anfragen an ein oder mehrere registrierte Ziele weiterleiten, z. B. EC2-Instances in Ihre Virtual Private Cloud (VPC).

Network Load Balancer

Führt das Routing und den Lastenausgleich auf Transportebene (TCP/UDP-Layer-4) basierend auf extrahierten Adressinformationen aus dem Layer-4-Header durch. Network Load Balancers können Datenverkehrsspitzen verarbeiten, die Quell-IP-Adresse des Clients beibehalten und eine feste IP für die Nutzungsdauer des Load Balancers verwenden.

Gateway Load Balancer

Verteilt den Datenverkehr an eine Flotte von Appliance-Instances. Bietet Skalierung, Verfügbarkeit und Einfachheit für virtuelle Appliances von Drittanbietern, wie Firewalls, Eindringungserkennungs- und -präventionssysteme und andere Appliances. Gateway Load Balancer arbeiten mit virtuellen Appliances, die das GENEVE-Protokoll unterstützen. Zusätzliche technische Integration ist erforderlich. Bitte konsultieren Sie daher das Benutzerhandbuch, bevor Sie einen Gateway Load Balancer auswählen.

Classic Load Balancer

Routen und Lastenausgleich entweder auf der Transportschicht (TCP/SSL), or at the application layer (HTTP/HTTPS).

Weitere Informationen zu den verschiedenen verfügbaren Load Balancer-Typen finden Sie in den folgenden Ressourcen:

- [Was ist Elastic Load Balancing?](#)
- [Was ist ein Application Load Balancer?](#)
- [Was ist ein Network Load Balancer?](#)
- [Was ist ein Gateway Load Balancer?](#)
- [Was ist ein Classic Load Balancer?](#)

Bereiten Sie das Anhängen eines Elastic Load Balancing Balancing-Load Balancers vor

Bevor Sie Ihrer Auto Scaling Scaling-Gruppe einen Elastic Load Balancing Load Balancer hinzufügen, müssen Sie die folgenden Voraussetzungen erfüllen:

- Sie müssen bereits den Load Balancer und die Zielgruppe erstellt haben, die für die Weiterleitung des Datenverkehrs an Ihre Auto Scaling Scaling-Gruppe verwendet werden.

Es gibt zwei Möglichkeiten, den Load Balancer und die Zielgruppe zu erstellen:

- Elastic Load Balancing verwenden — Folgen Sie den Verfahren in der Elastic Load Balancing Balancing-Dokumentation, um den Load Balancer und die Zielgruppe zu erstellen und zu konfigurieren, bevor Sie die Auto Scaling Scaling-Gruppe erstellen. Den Schritt zur Registrierung Ihrer Amazon EC2-Instances überspringen. Amazon EC2 Auto Scaling kümmert sich automatisch um die Registrierung (und Deregistrierung) von Instances, wenn Sie Ihrer Auto Scaling Scaling-Gruppe eine Zielgruppe zuordnen. Weitere Informationen finden Sie unter [Erste Schritte mit Elastic Load Balancing](#) im Elastic Load Balancing-Benutzerhandbuch.
- Verwenden von Amazon EC2 Auto Scaling — Erstellen, konfigurieren und verknüpfen Sie den Load Balancer und die Zielgruppe mit einer Basiskonfiguration von der Amazon EC2 Auto Scaling Scaling-Konsole aus. Weitere Informationen finden Sie unter [Konfigurieren Sie einen Application Load Balancer oder Network Load Balancer von der Konsole aus](#).
- Bevor Sie einen Load Balancer erstellen, sollten Sie wissen, welche Art von Load Balancer Sie benötigen. Weitere Informationen finden Sie unter [Arten von Elastic Load Balancing](#).
- Der Load Balancer und seine Zielgruppe müssen sich in derselben AWS-Konto VPC und Region wie Ihre Auto Scaling Scaling-Gruppe befinden.
- Die Zielgruppe muss den Zieltyp `instance` aufweisen. Sie können keinen Zieltyp von `ip` angeben, wenn Sie eine Auto-Scaling-Gruppe verwenden.
- Wenn die Startvorlage für Ihre Auto Scaling Scaling-Gruppe nicht die richtige Sicherheitsgruppe enthält, um den erforderlichen eingehenden Datenverkehr vom Load Balancer zuzulassen, müssen Sie die Startvorlage aktualisieren. Die empfohlenen Regeln hängen vom Typ des Load Balancers und den Arten von Backends ab, die der Load Balancer verwendet. Um beispielsweise Datenverkehr an Webserver weiterzuleiten, lassen Sie den eingehenden HTTP-Zugriff auf Port 80 vom Load Balancer zu. Bestehende Instances werden nicht mit den neuen Einstellungen aktualisiert, wenn die Startvorlage geändert wird. Um bestehende Instances zu aktualisieren, können Sie eine Instance-Aktualisierung starten, um die Instances zu ersetzen. Weitere

Informationen finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).

- Die Sicherheitsgruppen in der Startvorlage müssen außerdem den Zugriff vom Load Balancer auf den richtigen Port zulassen, damit Elastic Load Balancing seine Integritätsprüfungen durchführen kann.
- Bei der Bereitstellung virtueller Appliances hinter einem Gateway Load Balancer muss das Amazon Machine Image (AMI) in der Startvorlage die ID eines AMI angeben, das das GENEVE-Protokoll unterstützt, damit die Auto Scaling Scaling-Gruppe Datenverkehr mit einem Gateway Load Balancer austauschen kann. Außerdem müssen die Sicherheitsgruppen in der Startvorlage UDP-Verkehr auf Port 6081 zulassen.

Tip

Wenn Sie Bootstrapping-Skripte haben, deren Fertigstellung eine Weile dauert, können Sie Ihrer Auto-Scaling-Gruppe optional einen Start-Lebenszyklus-Hook hinzufügen, um die Registrierung von Instances hinter dem Load Balancer zu verzögern, bevor Ihre Bootstrap-Skripte erfolgreich abgeschlossen wurden und die Anwendungen auf den Instances bereit sind, Datenverkehr zu akzeptieren. Sie können keinen Lebenszyklus-Hook hinzufügen, wenn Sie zum ersten Mal eine Auto-Scaling-Gruppe in der Amazon-EC2-Auto-Scaling-Konsole erstellen. Sie können jedoch einen Lifecycle-Hook hinzufügen, nachdem die Gruppe erstellt wurde. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks bei Amazon EC2 Auto Scaling](#).

Konfigurieren Sie Integritätsprüfungen für Ziele

Sie können Integritätsprüfungen für Ihre Ziele konfigurieren, die bei einem Elastic Load Balancing Load Balancer registriert sind, um sicherzustellen, dass sie den Datenverkehr ordnungsgemäß verarbeiten können. Die spezifischen Schritte variieren je nach Art des Load Balancers, den Sie verwenden. Weitere Informationen finden Sie in den folgenden Ressourcen:

- Application Load Balancer — Informationen zu den [Zustandsprüfungen für Ihre Zielgruppen](#) finden Sie im Benutzerhandbuch für Application Load Balancer.
- Network Load Balancer — Weitere Informationen finden Sie im Benutzerhandbuch für Network Load Balancer unter Gesundheitschecks für [Ihre Zielgruppen](#).

- Gateway Load Balancer — Weitere Informationen finden Sie im Benutzerhandbuch für Gateway Load Balancer unter Gesundheitschecks für [Ihre Zielgruppen](#).
- Classic Load Balancer — Weitere Informationen finden [Sie unter Konfigurieren von Zustandsprüfungen für Ihren Classic Load Balancer](#) im Benutzerhandbuch für Classic Load Balancer.

Standardmäßig betrachtet Amazon EC2 Auto Scaling eine Instance nicht als fehlerhaft und ersetzt sie, wenn sie die Elastic Load Balancing Balancing-Zustandsprüfungen nicht besteht. Die Standard-Zustandsprüfungen für eine Auto-Scaling-Gruppe sind ausschließlich EC2-Zustandsprüfungen. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Damit Amazon EC2 Auto Scaling Instances ersetzen kann, die von Elastic Load Balancing als fehlerhaft gemeldet wurden, können Sie Ihre Auto Scaling Scaling-Gruppe so konfigurieren, dass sie Elastic Load Balancing Health Checks verwendet. Auf diese Weise betrachtet Amazon EC2 Auto Scaling die Instance als fehlerhaft, wenn sie entweder die EC2-Zustandsprüfungen oder die Elastic Load Balancing Balancing-Zustandsprüfungen nicht besteht. Wurden einer Gruppe mehrere Load Balancer-Zielgruppen oder Classic Load Balancer hinzugefügt, müssen alle melden, dass die Instance fehlerfrei ist, damit die Instance als fehlerfrei eingestuft wird. Wenn eine davon eine Instance als fehlerhaft meldet, ersetzt die Auto-Scaling-Gruppe die Instance, selbst dann, wenn sie von anderen als fehlerfrei gemeldet wird.

Informationen darüber, wie Sie diese Integritätsprüfungen für Ihre Auto Scaling Scaling-Gruppe aktivieren, finden Sie unter [Fügen Sie Ihrer Auto Scaling Scaling-Gruppe einen Elastic Load Balancing Load Balancer hinzu](#).

Note

Um sicherzustellen, dass diese Zustandsprüfungen so schnell wie möglich beginnen, stellen Sie sicher, dass die Kulanfrist für die Integritätsprüfung Ihrer Gruppe nicht zu hoch, sondern hoch genug ist, damit Ihre Elastic Load Balancing Balancing-Zustandsprüfungen feststellen können, ob ein Ziel für die Bearbeitung von Anfragen verfügbar ist. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).

Fügen Sie Ihrer Auto Scaling Scaling-Gruppe einen Elastic Load Balancing Load Balancer hinzu

In diesem Thema wird beschrieben, wie Sie einen Elastic Load Balancing Load Balancer an eine Auto Scaling Scaling-Gruppe anhängen. Außerdem wird beschrieben, wie Elastic Load Balancing Health Checks aktiviert werden, damit Amazon EC2 Auto Scaling Instances ersetzen kann, die Elastic Load Balancing als fehlerhaft meldet.

Standardmäßig ersetzt Amazon EC2 Auto Scaling nur Instances, die aufgrund von Amazon EC2-Zustandsprüfungen nicht zustandsbehaftet oder nicht erreichbar sind. Wenn Sie Elastic Load Balancing Health Checks aktivieren, kann Amazon EC2 Auto Scaling eine laufende Instance ersetzen, falls einer der Elastic Load Balancing Load Balancer, die Sie der Auto Scaling-Gruppe zuordnen, sie als fehlerhaft meldet.

Ein Tutorial zum Hinzufügen eines Application Load Balancer zu Ihrer Auto Scaling Scaling-Gruppe finden Sie unter: [Tutorial: Einrichten einer skalierten Anwendung mit Load Balancing](#)

Important

Bevor Sie fortfahren, müssen Sie alle im vorherigen Abschnitt genannten [Voraussetzungen](#) erfüllen.

Inhalt

- [Fügen Sie eine Zielgruppe oder einen Classic Load Balancer hinzu](#)
- [Eine Zielgruppe oder einen Classic Load Balancer abtrennen](#)

Fügen Sie eine Zielgruppe oder einen Classic Load Balancer hinzu

Wenn Sie eine Auto Scaling Scaling-Gruppe erstellen oder aktualisieren, können Sie eine oder mehrere Zielgruppen oder Classic Load Balancer anhängen. Wenn Sie einen Application Load Balancer, Network Load Balancer oder Gateway Load Balancer anhängen, fügen Sie eine Zielgruppe und nicht den Load Balancer selbst hinzu.

Befolgen Sie die Schritte in diesem Abschnitt, um die Konsole für Folgendes zu verwenden:

- Einer Auto Scaling Scaling-Gruppe eine Zielgruppe oder einen Classic Load Balancer zuordnen

- Schalten Sie die Integritätsprüfungen für Elastic Load Balancing ein

So fügen Sie beim Erstellen einer neuen Auto-Scaling-Gruppe einen vorhandenen Load Balancer hinzu

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben auf dem Bildschirm die aus, in der AWS-Region Sie Ihren Load Balancer erstellt haben.
3. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
4. Wählen Sie in den Schritten 1 und 2 die gewünschten Optionen aus und fahren Sie mit Schritt 3: Konfigurieren von erweiterten Optionen fort.
5. Für Load Balancing, wählen Sie An einen vorhandenen Load Balancer anhängen aus.
6. Bei An einen vorhandenen Load Balancer anhängen führen Sie im einen der folgenden Schritte aus:
 - a. Gehen Sie für Application Load Balancers, Network Load Balancers und Gateway Load Balancers folgendermaßen vor:

Klicken Sie auf Aus Ihren Zielgruppen für Load Balancer auswählen und wählen Sie dann im Feld Vorhandene Zielgruppen für Load Balancer eine Zielgruppe aus.
 - b. Für Classic Load Balancer:

Klicken Sie auf Auswählen aus Classic Load Balancers und wählen Sie dann Ihren Load Balancer im Feld Classic Load Balancer aus.
7. (Optional) Wählen Sie für Zustandsprüfungen und Zusätzliche Zustandsprüfungstypen die Option Elastic Load Balancing-Zustandsprüfungen aktivieren aus.
8. (Optional) Geben Sie unter Karenzzeit für die Zustandsprüfung die Zeit in Sekunden ein. So lange muss Amazon EC2 Auto Scaling warten, bevor der Zustand einer Instance überprüft wird nachdem Sie den Zustand InService erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).
9. Fahren Sie mit dem Erstellen der Auto-Scaling-Gruppe fort. Ihre Instances werden automatisch beim Load Balancer registriert, nachdem die Auto-Scaling-Gruppe erstellt wurde.

So fügen Sie einen vorhandenen Load Balancer an Ihre Auto-Scaling-Gruppe nach ihrer Erstellung an

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Integrationen die Optionen Load Balancing und Bearbeiten aus.
4. Führen Sie unter Load balancing (Lastenausgleich) eine der folgenden Aktionen aus:
 - a. Für Zielgruppen für Application, Network oder Gateway Load Balancer wählen Sie das entsprechende Kontrollkästchen und wählen eine Zielgruppe aus.
 - b. Für Classic Load Balancer wählen Sie das entsprechende Kontrollkästchen Ihren Load Balancer aus.
5. Wählen Sie Aktualisieren aus.

Wenn Sie mit dem Anhängen des Load Balancers fertig sind, können Sie optional die Integritätsprüfungen aktivieren, die ihn verwenden.

So aktivieren Sie die Elastic Load Balancing Health Checks

1. Wählen Sie auf der Registerkarte Details die Option Zustandsprüfungen, Bearbeiten aus.
2. Wählen Sie für Zustandsprüfungen und Zusätzliche Zustandsprüfungstypen die Option Elastic Load Balancing-Zustandsprüfungen aktivieren aus.
3. Geben Sie unter Frist für Zustandsprüfungen die Zeit in Sekunden ein. So lange muss Amazon EC2 Auto Scaling warten, bevor der Zustand einer Instance überprüft wird nachdem Sie den Zustand InService erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartefrist für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).
4. Wählen Sie Aktualisieren aus.

Note

Sie können den Status des Load Balancers überwachen, während er angefügt wird, indem Sie die AWS CLI verwenden. Wenn Amazon EC2 Auto Scaling die Instances erfolgreich registriert hat und mindestens eine registrierte Instance die Zustandsprüfungen besteht, erhalten Sie den Status von InService. Weitere Informationen finden Sie unter [Überprüfen des Anhangsstatus Ihres Load Balancers](#).

Eine Zielgruppe oder einen Classic Load Balancer abtrennen

Wird der Load Balancer nicht mehr benötigt, führen Sie die folgenden Schritte aus, um ihn von der Auto-Scaling-Gruppe zu trennen.

Trennen Sie einen Load Balancer wie folgt von einer Gruppe:

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Load Balancing, Bearbeiten.
4. Führen Sie unter Load balancing (Lastenausgleich) eine der folgenden Aktionen aus:
 - a. Wählen Sie für Application, Network oder Gateway Load Balancer-Zielgruppe das Löschsymbol (X) neben der Zielgruppe aus.
 - b. Wählen Sie unter Classic Load Balancer das Löschsymbol (X) neben dem Load Balancer aus.
5. Wählen Sie Aktualisieren aus.

Wenn Sie mit dem Trennen der Zielgruppe fertig sind, können Sie die Elastic Load Balancing Health Checks deaktivieren.

So deaktivieren Sie die Integritätsprüfungen von Elastic Load Balancing

1. Wählen Sie auf der Registerkarte Details die Option Zustandsprüfungen, Bearbeiten aus.

2. Deaktivieren Sie für Integritätsprüfungen und Zusätzliche Zustandsprüfungstypen die Option Elastic Load Balancing Balancing-Zustandsprüfungen aktivieren.
3. Wählen Sie Aktualisieren aus.

Konfigurieren Sie einen Application Load Balancer oder Network Load Balancer von der Konsole aus

Gehen Sie wie folgt vor, um beim Erstellen der Auto-Scaling-Gruppe einen Application Load Balancer oder einen Network Load Balancer zu erstellen und anzufügen.

Beim Erstellen einer neuen Auto-Scaling-Gruppe einen vorhandenen Load Balancer erstellen oder hinzufügen

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
3. Wählen Sie in den Schritten 1 und 2 die gewünschten Optionen aus und fahren Sie mit Schritt 3: Konfigurieren von erweiterten Optionen fort.
4. Für Load Balancing wählen Sie An einen neuen Load Balancer anfügen aus.
 - a. Bei An einen neuen Load Balancer anfügen wählen Sie für Load Balancer-Typ aus, ob ein Application Load Balancer oder Network Load Balancer erstellt werden soll.
 - b. Für Load Balancer-Name geben Sie einen Namen für den Load Balancer ein, oder behalten Sie den Standardnamen bei.
 - c. Für Load Balancer-Plan wählen Sie aus, ob ein öffentlicher mit dem Internet verbundener Load Balancer erstellt werden oder die Standardeinstellung für einen internen Load Balancer beibehalten werden soll.
 - d. Für Availability Zones und Subnetze wählen Sie für jede Availability Zone, in der Sie Ihre EC2-Instances starten möchten, das öffentliche Subnetz aus. (Diese werden ab Schritt 2 ausgefüllt.)
 - e. Für Listener und Routing aktualisieren Sie die Portnummer für Ihren Listener (falls erforderlich) und unter Standard-Routing wählen Sie Erstellen einer Zielgruppe aus. Alternativ können Sie eine vorhandene Zielgruppe aus der Dropdown-Liste auswählen.

- f. Wenn Sie im letzten Schritt Erstellen einer Zielgruppe ausgewählt haben, geben Sie für Name der neuen Zielgruppe einen Namen für die Zielgruppe ein, oder behalten Sie den Standardnamen bei.
 - g. Um Tags zu Ihrem Load Balancer hinzuzufügen, wählen Sie Add tag (Tag hinzufügen) und geben Sie einen Tag-Schlüssel und den Wert für jedes Tag an.
5. (Optional) Wählen Sie für Zustandsprüfungen und Zusätzliche Zustandsprüfungstypen die Option Elastic Load Balancing-Zustandsprüfungen aktivieren aus.
 6. (Optional) Geben Sie unter Karenzzeit für die Zustandsprüfung die Zeit in Sekunden ein. So lange muss Amazon EC2 Auto Scaling warten, bevor der Zustand einer Instance überprüft wird nachdem Sie den Zustand InService erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).
 7. Fahren Sie mit dem Erstellen der Auto-Scaling-Gruppe fort. Ihre Instances werden automatisch beim Load Balancer registriert, nachdem die Auto-Scaling-Gruppe erstellt wurde.

Note

Nachdem Sie die Auto-Scaling-Gruppe erstellt haben, können Sie mit der Elastic Load Balancing-Konsole zusätzliche Listener erstellen. Dies ist nützlich, wenn Sie einen Listener mit einem sicheren Protokoll wie HTTPS oder einem UDP-Listener erstellen müssen. Sie können vorhandenen Load Balancern weitere Listener hinzufügen, sofern Sie unterschiedliche Ports verwenden.

Überprüfen des Anhangsstatus Ihres Load Balancers

Nachdem Sie einen Load Balancer hinzufügen, wird er in den Status Adding versetzt, solange er die Instances der Gruppe registriert. Wenn alle Instances der Gruppe angemeldet sind, wird sie in den Status Added versetzt. Besteht zumindest eine angemeldete Instance die Zustandsprüfungen, wird er in den Status InService versetzt. Wurde der Load Balancer in den Status InService versetzt, kann Amazon EC2 Auto Scaling alle Instances beenden und ersetzen, die als fehlerhaft gemeldet werden. Besteht keine angemeldete Instance die Zustandsprüfungen (beispielsweise aufgrund einer falsch konfigurierten Zustandsprüfung), wird der Load Balancer nicht in den Status InService versetzt. Amazon EC2 Auto Scaling beendet und ersetzt die Instances nicht.

Wenn Sie einen Load Balancer trennen, wird er in den Status Removing versetzt, solange er die Instances der Gruppe abmeldet. Nach der Abmeldung werden die Instances weiterhin ausgeführt.

Standardmäßig ist Connection Draining für Application Load Balancer, Network Load Balancer und Gateway Load Balancer aktiviert. Ist Connection Draining aktiviert, wartet Elastic Load Balancing darauf, dass aktive Anforderungen abgeschlossen werden oder das maximale Zeitlimit abgelaufen ist (je nachdem, was zuerst eintritt), bevor die Instances abgemeldet werden.

Sie können den Status der Anlage überprüfen, indem Sie AWS Command Line Interface (AWS CLI) oder verwenden. AWS SDKs Sie können den Status des Anhangs nicht von der Konsole aus überprüfen.

AWS CLI Um den Status des Anhangs zu überprüfen

Der folgende [describe-traffic-sources](#) Befehl gibt den Anhangsstatus aller Verkehrsquellen für die angegebene Auto Scaling Scaling-Gruppe zurück.

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

Das Beispiel gibt den ARN von Elastic-Load-Balancing zurück, das an die Auto-Scaling-Gruppe angehängt ist, zusammen mit dem Anhangsstatus der Zielgruppe im Element State.

```
{
  "TrafficSources": [
    {
      "Identifier": "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456",
      "State": "InService",
      "Type": "elbv2"
    }
  ]
}
```

Fügen Sie eine Availability Zone hinzu

Sie können die Vorteile der Sicherheit und Zuverlässigkeit geografischer Redundanz nutzen, indem Sie Ihre Auto-Scaling-Gruppe auf mehrere Availability Zones der Region, in der Sie arbeiten, verteilen und dann einen Load Balancer zum Verteilen des eingehenden Datenverkehrs auf diese Availability Zones hinzufügen.

Wenn eine Availability Zone fehlerhaft oder nicht verfügbar ist, startet Amazon EC2 Auto Scaling neue Instances in einer nicht betroffenen Availability Zone. Wenn die fehlerhafte Availability Zone

meldet, dass sie wieder fehlerfrei ist, verteilt Amazon EC2 Auto Scaling die Anwendungs-Instances automatisch gleichmäßig auf alle Availability Zones für Ihre Auto-Scaling-Gruppe. Amazon EC2 Auto Scaling versucht dazu, in der Availability Zone mit den wenigsten Instances neue Instances zu starten. Scheitert dies jedoch, setzt Amazon EC2 Auto Scaling den Versuch in anderen Availability Zones fort, bis er erfolgreich ist.

Elastic Load Balancing erstellt einen Load Balancer-Knoten für jede Availability Zone, die Sie für den Load Balancer aktivieren. Wenn zonenübergreifendes Load Balancing für Ihren Load Balancer aktiviert ist, verteilt jeder Load Balancer-Knoten den Datenverkehr gleichmäßig auf die registrierten Ziele in allen aktivierten Availability Zones. Wenn zonenübergreifendes Load Balancing deaktiviert ist, verteilt jeder Load Balancer-Knoten Anfragen gleichmäßig nur auf die registrierten Instances in seiner aktivierten Availability Zone.

Sie müssen mindestens eine Availability Zone angeben, wenn Sie Ihre Auto-Scaling-Gruppe erstellen. Sie können die Verfügbarkeit der skalierten Anwendung mit Lastenausgleich erweitern, indem Sie der Auto-Scaling-Gruppe ein Availability Zone hinzufügen und dann dem Load Balancer Zugriff auf sie gewähren (sofern der Load Balancer dies unterstützt).

Einschränkungen

Zur Aktualisierung der Availability Zones, die für Ihren Load Balancer aktiviert sind, müssen Sie die folgenden Einschränkungen kennen:

- Wenn Sie eine Availability Zone für Ihren Load Balancer aktivieren, geben Sie ein Subnetz aus dieser Availability Zone an. Beachten Sie, dass Sie höchstens ein Subnetz pro Availability Zone für Ihren Load Balancer aktivieren können.
- Für Load Balancer mit Internetverbindung müssen die von Ihnen für den Load Balancer angegebenen Subnetze über mindestens acht verfügbare IP-Adressen verfügen.
- Für Application Load Balancers müssen Sie Subnetze aus mindestens zwei Availability Zones angeben.
- Bei Network Load Balancern können Sie die aktivierten Availability Zones nicht deaktivieren, aber Sie können zusätzliche Zones aktivieren.
- Für Gateway Load Balancer können Sie die aktivierten Availability Zones nicht deaktivieren, aber Sie können zusätzliche aktivieren.

Gehen Sie wie folgt vor, um Ihre Auto-Scaling-Gruppe und Ihren Load Balancer auf ein Subnetz in einer zusätzlichen Availability Zone zu erweitern.

Hinzufügen einer Availability Zone

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Netzwerk, Bearbeiten.
4. In Subnetze wählen Sie das Subnetz aus, das der Availability Zone entspricht, das Sie der Auto-Scaling-Gruppe hinzufügen möchten.
5. Wählen Sie Aktualisieren aus.
6. Führen Sie die folgenden Schritte aus, um die Availability Zones für Ihren Load Balancer so zu aktualisieren, dass er dieselben Zonen wie Ihre Auto-Scaling-Gruppe verwendet:
 - a. Wählen Sie im Navigationsbereich unter LOAD BALANCING die Option Load Balancers aus.
 - b. Wählen Sie Ihren -Load Balancer.
 - c. Führen Sie eine der folgenden Aktionen aus:
 - Für Application und Network Load Balancer:
 1. Wählen Sie auf der Registerkarte Description (Beschreibung) für Availability Zones Edit subnets (Subnetze bearbeiten) aus.
 2. Klicken Sie auf der Seite Subnetze bearbeiten für Availability Zones auf das Kontrollkästchen für die hinzuzufügende Availability Zone. Wenn es nur ein Subnetz für diese Zone gibt, ist es ausgewählt. Wenn es mehr als ein Subnetz für diese Zone gibt, wählen Sie eines der Subnetze aus.
 - Für Classic Load Balancer in einer VPC:
 1. Wählen Sie auf der Registerkarte Instances die Option Edit Availability Zones.
 2. Wählen Sie auf der Seite Add and Remove Subnets (Subnetze hinzufügen und entfernen) unter Available subnets (Verfügbare Subnetze) für das hinzuzufügende Subnetz das Symbol „Hinzufügen“ (+) aus. Das Subnetz wird nach Selected subnets verschoben.
 - d. Wählen Sie Speichern.

Zugehörige Ressourcen

Amazon EC2 Auto Scaling gleicht Ihre Gruppe aus, wenn Sie Availability Zones ändern. Das heißt, dass einige Instances ersetzt und neu verteilt werden müssen. Weitere Informationen finden Sie unter [Beispiel: Aufteilen von Instances in mehrere Availability Zones](#).

Wenn Sie Ziele in Availability Zones registriert haben, die nicht für den Load Balancer aktiviert sind, leitet der Load Balancer keinen Traffic an diese Ziele weiter. Weitere Informationen finden Sie unter [Funktionsweise von Elastic Load Balancing](#) im Benutzerhandbuch für Elastic Load Balancing.

Entfernen einer Availability Zone

Gehen Sie wie folgt vor, um eine Availability Zone aus Ihrer Auto-Scaling-Gruppe und Ihrem Load Balancer zu entfernen.

Entfernen einer Availability Zone

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Netzwerk, Bearbeiten.
4. In Subnetze wählen Sie das Lösch-Symbol (X) für das Subnetz aus, das der Availability Zone entspricht, das Sie aus der Auto-Scaling-Gruppe entfernen möchten. Wenn es mehr als ein Subnetz für diese Zone gibt, wählen Sie für jede Zone das Löschsymboll (X) aus.
5. Wählen Sie Aktualisieren aus.
6. Führen Sie die folgenden Schritte aus, um die Availability Zones für Ihren Load Balancer so zu aktualisieren, dass er dieselben Zonen wie Ihre Auto-Scaling-Gruppe verwendet:
 - a. Wählen Sie im Navigationsbereich unter LOAD BALANCING die Option Load Balancers aus.
 - b. Wählen Sie Ihren -Load Balancer.
 - c. Führen Sie eine der folgenden Aktionen aus:
 - Für Application Load Balancer:

1. Wählen Sie auf der Registerkarte Description (Beschreibung) für Availability Zones Edit subnets (Subnetze bearbeiten) aus.
 2. Deaktivieren Sie auf der Seite Subnetze bearbeiten für Availability Zones das Kontrollkästchen, um das Subnetz für die Availability Zone zu entfernen.
- Für Classic Load Balancer in einer VPC:
 1. Wählen Sie auf der Registerkarte Instances die Option Edit Availability Zones.
 2. Entfernen Sie auf der Seite Add and Remove Subnets (Subnetze hinzufügen und entfernen) unter Available subnets (Verfügbare Subnetze) das Subnetz über dessen Löschsymbol (-). Das Subnetz wird zu Available Subnets verschoben.
- d. Wählen Sie Speichern.

Trennen Sie eine Zielgruppe oder einen Classic Load Balancer von Ihrer Auto Scaling Scaling-Gruppe

Wird der Load Balancer nicht mehr benötigt, führen Sie die folgenden Schritte aus, um ihn von der Auto-Scaling-Gruppe zu trennen.

Trennen Sie einen Load Balancer wie folgt von einer Gruppe:

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite Auto Scaling groups (Auto-Scaling-Gruppen) wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option Load Balancing, Bearbeiten.
4. Führen Sie unter Load balancing (Lastenausgleich) eine der folgenden Aktionen aus:
 - a. Wählen Sie für Application, Network oder Gateway Load Balancer-Zielgruppe das Löschsymbol (X) neben der Zielgruppe aus.
 - b. Wählen Sie unter Classic Load Balancer das Löschsymbol (X) neben dem Load Balancer aus.
5. Wählen Sie Aktualisieren aus.

Wenn Sie mit dem Trennen der Zielgruppe fertig sind, können Sie die Elastic Load Balancing Health Checks deaktivieren.

So deaktivieren Sie die Integritätsprüfungen von Elastic Load Balancing

1. Wählen Sie auf der Registerkarte Details die Option Zustandsprüfungen, Bearbeiten aus.
2. Deaktivieren Sie für Integritätsprüfungen und Zusätzliche Zustandsprüfungstypen die Option Elastic Load Balancing Balancing-Zustandsprüfungen aktivieren.
3. Wählen Sie Aktualisieren aus.

Beispiele für die Arbeit mit Elastic Load Balancing unter Verwendung der AWS CLI

Verwenden Sie AWS Command Line Interface (AWS CLI), um Load Balancer und Zielgruppen anzuhängen, zu trennen und zu beschreiben, Elastic Load Balancing Health Checks hinzuzufügen und zu entfernen und zu ändern, welche Availability Zones aktiviert sind.

Dieses Thema zeigt Beispiele für AWS CLI Befehle, die allgemeine Aufgaben für Amazon EC2 Auto Scaling ausführen.

Important

Weitere Befehlsbeispiele finden Sie unter [aws elbv2](#) und [aws elb](#) in der AWS CLI - Befehlsreferenz.

Inhalt

- [Hängen Sie Ihre Zielgruppe oder Ihren Classic Load Balancer an.](#)
- [Beschreiben Sie Ihre Zielgruppen oder Classic Load Balancers.](#)
- [Hinzufügen von Elastic Load Balancing-Zustandsprüfungen](#)
- [Ändern Ihrer Availability Zones](#)
- [Trennen Sie Ihre Zielgruppe oder Ihren Classic Load Balancer.](#)
- [Entfernen von Elastic Load Balancing-Zustandsprüfungen](#)
- [Legacybefehle](#)

Hängen Sie Ihre Zielgruppe oder Ihren Classic Load Balancer an.

Verwenden Sie den folgenden [create-auto-scaling-group](#) Befehl, um eine Auto Scaling Scaling-Gruppe zu erstellen und gleichzeitig eine Zielgruppe anzuhängen, indem Sie ihren Amazon-Ressourcennamen (ARN) angeben. Die Zielgruppe kann einem Application Load Balancer, einem Network Load Balancer oder einem Gateway Load Balancer zugeordnet werden.

Ersetzen Sie die Beispielwerte für `--auto-scaling-group-name`, `--vpc-zone-identifier`, `--min-size` und `--max-size`. Ersetzen Sie für die Option `--launch-template` `my-launch-template` und `1` durch den Namen und die Version einer Startvorlage für Ihre Auto-Scaling-Gruppe. Für die Option `--traffic-sources` ersetzen Sie den Beispiel-ARN durch den ARN einer Zielgruppe für einen Application Load Balancer, Network Load Balancer oder Gateway Load Balancer.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
  --min-size 1 --max-size 5 \  
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/12345678EXAMPLE1"
```

Verwenden Sie den [attach-traffic-sources](#) Befehl, um der Auto Scaling Scaling-Gruppe weitere Zielgruppen hinzuzufügen, nachdem sie erstellt wurde.

Mit dem folgenden Befehl fügen Sie derselben Gruppe eine weitere Zielgruppe hinzu.

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \  
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/12345678EXAMPLE2"
```

Um Ihrer Gruppe einen Classic Load Balancer hinzuzufügen, geben Sie alternativ die Optionen `--traffic-sources` und `--type` an, wenn Sie `create-auto-scaling-group` oder `attach-traffic-sources` verwenden, wie im folgenden Beispiel. Ersetzen Sie `my-classic-load-balancer` durch den Namen eines Classic Load Balancer. Geben Sie für die Option `--type` einen Wert von `elb` an.

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

Beschreiben Sie Ihre Zielgruppen oder Classic Load Balancers.

Verwenden Sie den folgenden [describe-traffic-sources](#) Befehl, um die Load Balancer oder Zielgruppen zu beschreiben, die Ihrer Auto Scaling Scaling-Gruppe zugeordnet sind. Ersetzen Sie *my-asg* durch den Namen Ihrer Gruppe.

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

Das Beispiel gibt den ARN der Elastic Load Balancing-Zielgruppen zurück, die Sie der Auto-Scaling-Gruppe hinzugefügt haben.

```
{
  "TrafficSources": [
    {
      "Identifier": "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/12345678EXAMPLE1",
      "State": "InService",
      "Type": "elbv2"
    },
    {
      "Identifier": "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/12345678EXAMPLE2",
      "State": "InService",
      "Type": "elbv2"
    }
  ]
}
```

Eine Erklärung des State-Felds in der Ausgabe finden Sie unter [Überprüfen des Anhangsstatus Ihres Load Balancers](#).

Hinzufügen von Elastic Load Balancing-Zustandsprüfungen

Um Elastic Load Balancing Health Checks zu den Integritätsprüfungen hinzuzufügen, die Ihre Auto Scaling Scaling-Gruppe an Instances durchführt, verwenden Sie den folgenden [update-auto-scaling-group](#) Befehl und geben Sie **ELB** als Wert für die `--health-check-type` Option an. Ersetzen Sie *my-asg* durch den Namen Ihrer Gruppe.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "ELB"
```

Neue Instances benötigen oft Zeit für eine kurze Aufwärmphase, bevor sie eine Zustandsprüfung bestehen können. Wenn die Übergangszeit nicht ausreichend Aufwärmzeit bietet, scheinen die Instances möglicherweise nicht bereit zu sein, Traffic zu verarbeiten. Amazon EC2 Auto Scaling kann diese Instances als fehlerhaft betrachten und ersetzen.

Verwenden Sie zum Aktualisieren der Frist für die Zustandsprüfung die Option `--health-check-grace-period`, wenn Sie `update-auto-scaling-group` verwenden, wie im folgenden Beispiel.

~~300~~ Ersetzen Sie es durch die Anzahl der Sekunden, um neue Instances in Betrieb zu halten, bevor sie beendet werden, wenn sich herausstellt, dass sie fehlerhaft sind.

```
--health-check-grace-period 300
```

Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Ändern Ihrer Availability Zones

Für das Ändern Ihrer Availability Zones gelten einige Einschränkungen, die Sie kennen sollten. Weitere Informationen finden Sie unter [Fügen Sie eine Availability Zone hinzu](#).

So ändern Sie die Availability Zones für einen Application Load Balancer oder Network Load Balancer

1. Bevor Sie die Availability Zones des Load Balancers ändern, empfiehlt es sich, zunächst die Availability Zones der Auto-Scaling-Gruppe zu aktualisieren, um sicherzustellen, dass Ihre Instance-Typen in den angegebenen Zonen verfügbar sind.

Verwenden Sie den folgenden [update-auto-scaling-group](#) Befehl, um die Availability Zones für Ihre Auto Scaling Scaling-Gruppe zu aktualisieren. Ersetzen Sie das Beispielsubnetz IDs durch das IDs der Subnetze in den Availability Zones, die aktiviert werden sollen. Die angegebenen Subnetze ersetzen die zuvor aktivierten Subnetze. Ersetzen Sie `my-asg` durch den Namen Ihrer Gruppe.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--vpc-zone-identifier "subnet-41767929,subnet-cb663da2,subnet-8360a9e7"
```

2. Verwenden Sie den folgenden [describe-auto-scaling-groups](#) Befehl, um zu überprüfen, ob die Instances in den neuen Subnetzen gestartet wurden. Wenn die Instances gestartet wurden, wird eine Liste der Instances und deren Status angezeigt. Ersetzen Sie `my-asg` durch den Namen Ihrer Gruppe.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. Verwenden Sie den folgenden Befehl [set-subnets](#), um die Subnetze für Ihren Load Balancer anzugeben. Ersetzen Sie das Beispielsubnetz IDs durch das IDs der Subnetze in den Availability Zones, um es zu aktivieren. Sie können nur ein Subnetz pro Availability Zone angeben. Die angegebenen Subnetze ersetzen die zuvor aktivierten Subnetze. Ersetzen Sie *my-lb-arn* durch den ARN Ihres Load Balancers.

```
aws elbv2 set-subnets --load-balancer-arn my-lb-arn \  
--subnets subnet-41767929 subnet-cb663da2 subnet-8360a9e7
```

Ändern der Availability Zones für einen Classic Load Balancer

1. Bevor Sie die Availability Zones des Load Balancers ändern, empfiehlt es sich, zunächst die Availability Zones der Auto-Scaling-Gruppe zu aktualisieren, um sicherzustellen, dass Ihre Instance-Typen in den angegebenen Zonen verfügbar sind.

Verwenden Sie den folgenden [update-auto-scaling-group](#) Befehl, um die Availability Zones für Ihre Auto Scaling Scaling-Gruppe zu aktualisieren. Ersetzen Sie das Beispielsubnetz IDs durch das IDs der Subnetze in den Availability Zones, die aktiviert werden sollen. Die angegebenen Subnetze ersetzen die zuvor aktivierten Subnetze. Ersetzen Sie *my-asg* durch den Namen Ihrer Gruppe.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--vpc-zone-identifier "subnet-41767929,subnet-cb663da2"
```

2. Verwenden Sie den folgenden [describe-auto-scaling-groups](#) Befehl, um zu überprüfen, ob die Instances in den neuen Subnetzen gestartet wurden. Wenn die Instances gestartet wurden, wird eine Liste der Instances und deren Status angezeigt. Ersetzen Sie *my-asg* durch den Namen Ihrer Gruppe.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. Verwenden Sie den folgenden Befehl [attach-load-balancer-to-subnets](#), um eine neue Availability Zone für Ihren Classic Load Balancer zu aktivieren. Ersetzen Sie zum Aktivieren die Beispiel-Subnetz-ID durch die ID des Subnetzes für die Availability Zone. Ersetzen Sie *my-lb* durch den Namen Ihres Load Balancers.

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-lb \  
--subnets subnet-cb663da2
```

Verwenden Sie den folgenden [detach-load-balancer-from](#) Befehl `-subnets`, um eine [Availability Zone zu deaktivieren](#). Ersetzen Sie zum Deaktivieren die Beispiel-Subnetz-ID durch die ID des Subnetzes für die Availability Zone. Ersetzen Sie *my-lb* durch den Namen Ihres Load Balancers.

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-lb \  
--subnets subnet-8360a9e7
```

Trennen Sie Ihre Zielgruppe oder Ihren Classic Load Balancer.

Der folgende [detach-traffic-sources](#) Befehl trennt eine Zielgruppe von Ihrer Auto Scaling Scaling-Gruppe, wenn Sie sie nicht mehr benötigen.

Ersetzen Sie für die Option `--auto-scaling-group-name` *my-asg* durch den Namen Ihrer Gruppe. Für die Option `--traffic-sources` ersetzen Sie den Beispiel-ARN durch den ARN einer Zielgruppe für einen Application Load Balancer, Network Load Balancer oder Gateway Load Balancer.

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \  
--traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-  
id:targetgroup/my-targets/1234567890123456"
```

Um einen Classic Load Balancer von Ihrer Gruppe zu trennen, geben Sie die Optionen `--traffic-sources` und `--type` an, wie im folgenden Beispiel. Ersetzen Sie *my-classic-load-balancer* durch den Namen eines Classic Load Balancer. Geben Sie für die Option `--type` einen Wert von **elb** an.

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

Entfernen von Elastic Load Balancing-Zustandsprüfungen

Um Elastic Load Balancing Health Checks aus Ihrer Auto Scaling Scaling-Gruppe zu entfernen, verwenden Sie den folgenden [update-auto-scaling-group](#) Befehl und geben Sie **EC2** als Wert für die `--health-check-type` Option an. Ersetzen Sie *my-asg* durch den Namen Ihrer Gruppe.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--health-check-type "EC2"
```

Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

Legacybefehle

Die folgenden Beispiele beschreiben, wie Sie Legacy-CLI-Befehle verwenden können, um Load Balancer und Zielgruppen anzufügen, zu trennen und zu beschreiben. Sie bleiben in diesem Dokument als Referenz für alle Kunden, die sie weiterhin verwenden möchten. Wir unterstützen weiterhin die alten CLI-Befehle, empfehlen jedoch, die neuen CLI-Befehle für „Trafficquellen“ zu verwenden, mit denen mehrere Datenverkehrsquellen-Typen angehängt und getrennt werden können. Sie können sowohl die Legacy-CLI-Befehle als auch die CLI-Befehle für „Trafficquellen“ in derselben Auto-Scaling-Gruppe verwenden.

Hängen Sie Ihre Zielgruppe oder Ihren Classic Load Balancer an (Legacy)

Fügen Sie Ihre Zielgruppe wie folgt hinzu:

Der folgende [create-auto-scaling-group](#) Befehl erstellt eine Auto Scaling Scaling-Gruppe mit einer angehängten Zielgruppe. Geben Sie den Amazon-Ressourcenname (ARN) einer Zielgruppe für einen Application Load Balancer, Network Load Balancer oder Gateway Load Balancer an.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
--launch-template LaunchTemplateName=my-launch-template,Version='1' \  
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
--target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-  
targets/1234567890123456" \  
--min-size 1 --max-size 5
```

Mit dem folgenden Befehl [attach-load-balancer-target-groups](#) wird eine Zielgruppe an eine bestehende Auto Scaling Scaling-Gruppe angehängt.

```
aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-name my-asg \  
--target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-  
targets/1234567890123456"
```

Anhängen Ihres Classic Load Balancers

Der folgende [create-auto-scaling-group](#) Befehl erstellt eine Auto Scaling Scaling-Gruppe mit einem angehängten Classic Load Balancer.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
  --launch-configuration-name my-launch-config \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
  --load-balancer-names "my-load-balancer" \  
  --min-size 1 --max-size 5
```

Der folgende [attach-load-balancers](#) Befehl fügt den angegebenen Classic Load Balancer einer vorhandenen Auto Scaling Scaling-Gruppe hinzu.

```
aws autoscaling attach-load-balancers --auto-scaling-group-name my-asg \  
  --load-balancer-names my-lb
```

Beschreiben Sie Ihre Zielgruppe oder Ihren Classic Load Balancer (Legacy)

Beschreiben von Zielgruppen

[Verwenden Sie den Befehl -groups, um die Zielgruppen zu beschreiben, die describe-load-balancer-target einer Auto Scaling Scaling-Gruppe zugeordnet sind.](#) Das folgende Beispiel listet die Zielgruppen für *my-asg* auf.

```
aws autoscaling describe-load-balancer-target-groups --auto-scaling-group-name my-asg
```

Beschreiben eines Classic Load Balancers

Verwenden Sie den [describe-load-balancers](#) Befehl, um die Classic Load Balancers zu beschreiben, die einer Auto Scaling Scaling-Gruppe zugeordnet sind. Das folgende Beispiel listet die Classic Load Balancers für auf. *my-asg*

```
aws autoscaling describe-load-balancers --auto-scaling-group-name my-asg
```

Trennen Sie Ihre Zielgruppe oder Ihren Classic Load Balancer (Legacy)

Trennen Sie eine Zielgruppe wie folgt:

Der folgende Befehl [detach-load-balancer-target-groups](#) trennt eine Zielgruppe von Ihrer Auto Scaling Scaling-Gruppe, wenn Sie sie nicht mehr benötigen.

```
aws autoscaling detach-load-balancer-target-groups --auto-scaling-group-name my-asg \  
  --load-balancer-names my-lb
```

```
--target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456"
```

Trennen eines Classic Load Balancers

Mit dem folgenden [detach-load-balancers](#) Befehl wird ein Classic Load Balancer von Ihrer Auto Scaling Scaling-Gruppe getrennt, wenn Sie ihn nicht mehr benötigen.

```
aws autoscaling detach-load-balancers --auto-scaling-group-name my-asg \  
--load-balancer-names my-lb
```

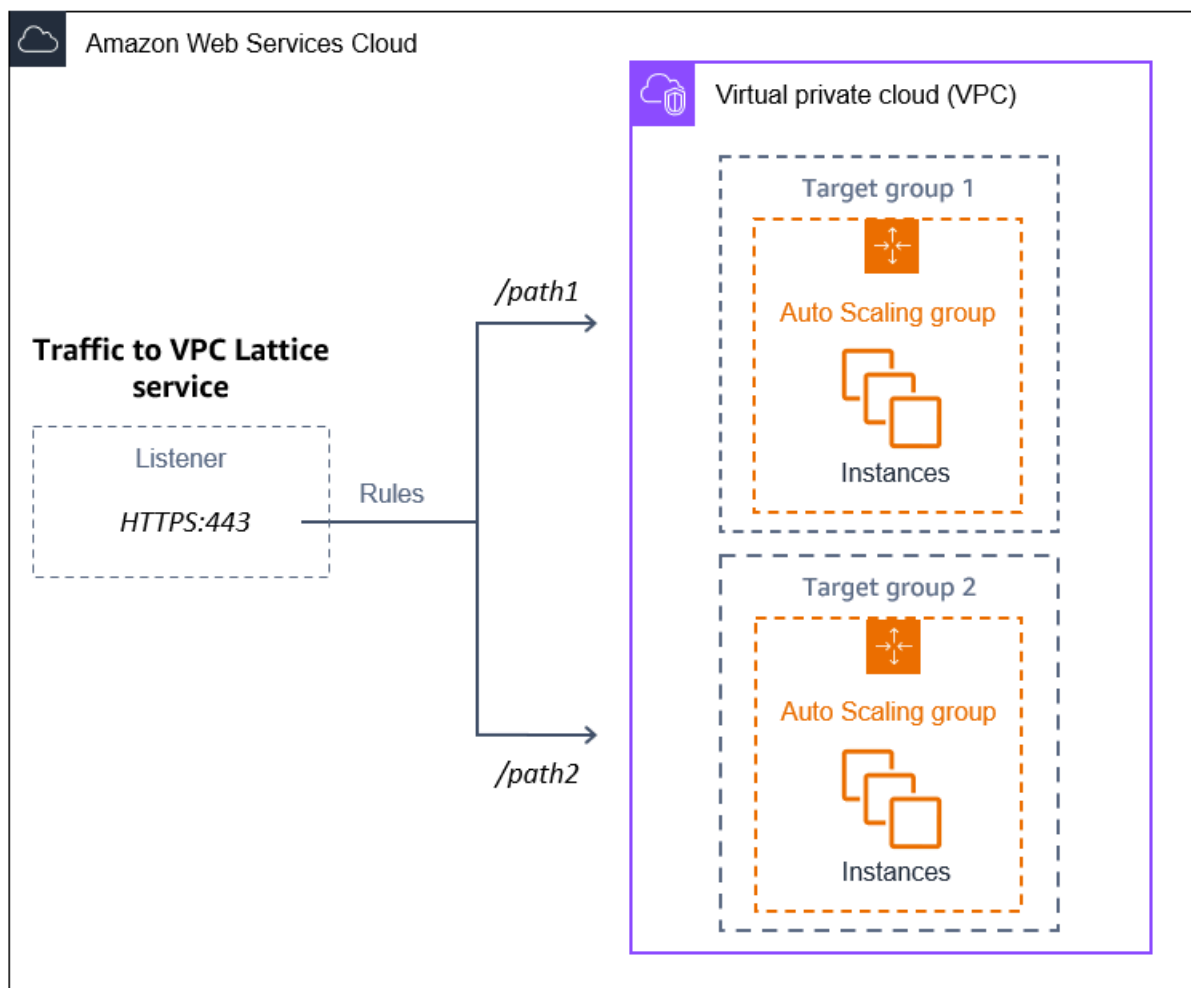
Steuern Sie den Verkehrsfluss mit einer Zielgruppe von VPC Lattice

Sie können Amazon VPC Lattice verwenden, um den Datenverkehr und die API-Aufrufe zwischen Ihren Anwendungen und Services zu verwalten, die auf separaten Ressourcen laufen, wie z. B. Auto Scaling-Gruppen oder Lambda-Funktionen. VPC Lattice ist ein Anwendungsnetzwerkdienst, mit dem Sie all Ihre Dienste über mehrere Konten und virtuelle private Clouds hinweg verbinden, sichern und überwachen können ()VPCs. Weitere Informationen zu VPC Lattice finden Sie unter [Was ist VPC Lattice?](#)

Um mit VPC Lattice zu beginnen, erstellen Sie zunächst die erforderlichen VPC Lattice-Ressourcen, die es den Ressourcen in einer mit einem Service-Netzwerk verbundenen VPC erlaubt, sich miteinander zu verbinden. Zu diesen Ressourcen gehören die Services, Listener, Listener-Regeln und Zielgruppen.

Um eine Auto Scaling-Gruppe mit einem VPC Lattice-Dienst zu verknüpfen, erstellen Sie eine Zielgruppe für den Dienst, die Anfragen an die nach Instance-ID registrierten Instances weiterleitet, und fügen Sie dem Dienst einen Listener hinzu, der Anfragen an die Zielgruppe sendet. Verbinden Sie dann die Zielgruppe mit Ihrer Auto-Scaling-Gruppe. Amazon EC2 Auto Scaling registriert die EC2-Instances automatisch als Ziele bei der Zielgruppe. Wenn Amazon EC2 Auto Scaling später eine Instance beenden muss, wird die Instance vor der Beendigung automatisch von der Zielgruppe abgemeldet.

Nachdem Sie die Zielgruppe hinzugefügt haben, ist sie der Einstiegspunkt für alle eingehenden Anfragen an Ihre Auto-Scaling-Gruppe. Wie das Beispiel im folgenden Diagramm zeigt, können eingehende Anfragen dann mithilfe der für einen VPC Lattice-Dienst angegebenen Listener-Regeln an die entsprechende Zielgruppe weitergeleitet werden.



Wenn der Datenverkehr durch VPC Lattice zu Ihrer Auto-Scaling-Gruppe geleitet wird, gleicht VPC Lattice die Anfragen unter den Instances in der Gruppe durch Round-Robin-Load Balancing aus. VPC Lattice kann auch den Zustand seiner registrierten Instances überwachen und den Datenverkehr nur an gesunde Instances weiterleiten.

Um Ihre Instances für eingehende Anfragen verfügbar zu halten, können Sie Ihrer Auto-Scaling-Gruppe optional VPC Lattice-Zustandsprüfungen hinzufügen. Wenn eine der EC2-Instances ausfällt, startet Ihre Auto Scaling-Gruppe automatisch eine neue Instance, um sie zu ersetzen. Das Verhalten der Zustandsprüfungen von VPC Lattice ähnelt dem Verhalten der Zustandsprüfungen des Elastic Load Balancing. Die Standard-Zustandsprüfungen für eine Auto-Scaling-Gruppe sind ausschließlich EC2-Zustandsprüfungen.

Weitere Informationen zu VPC Lattice finden Sie unter [Vereinfachen von Service-to-Service Konnektivität, Sicherheit und Überwachung mit Amazon VPC Lattice — Jetzt allgemein verfügbar im Blog](#). AWS

Inhalt

- [Vorbereiten des Hinzufügens einer VPC-Lattice-Zielgruppe an Ihre Auto-Scaling-Gruppe](#)
- [Hinzufügen einer VPC-Lattice-Zielgruppe zu Ihrer Auto-Scaling-Gruppe](#)
- [Überprüfen Sie den Anhangsstatus Ihrer VPC Lattice-Zielgruppe](#)

Vorbereiten des Hinzufügens einer VPC-Lattice-Zielgruppe an Ihre Auto-Scaling-Gruppe

Bevor Sie eine VPC Lattice-Zielgruppe mit Ihrer Auto-Scaling-Gruppe verbinden, müssen Sie die folgenden Voraussetzungen erfüllen:

- Sie müssen bereits ein VPC-Lattice-Dienstnetzwerk, einen Dienst, einen Listener und eine Zielgruppe erstellt haben. Weitere Informationen finden Sie unter den folgenden Themen im VPC Lattice Benutzerhandbuch:
 - [Servicenetze](#)
 - [Services](#)
 - [Listener](#)
 - [Zielgruppen](#)
- Die Zielgruppe muss sich in derselben AWS-Konto VPC und Region wie Ihre Auto Scaling Scaling-Gruppe befinden.
- Die Zielgruppe muss den Zieltyp `instance` aufweisen. Sie können keinen Zieltyp von `ip` angeben, wenn Sie eine Auto-Scaling-Gruppe verwenden.
- Sie benötigen ausreichende IAM-Berechtigungen, um die Zielgruppe an die Auto-Scaling-Gruppe anhängen zu können. Die folgende Beispielrichtlinie zeigt die mindestens erforderlichen Berechtigungen, die zum Anhängen und Trennen von Zielgruppen erforderlich sind.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:AttachTrafficSources",
        "autoscaling:DetachTrafficSources",
```

```
        "autoscaling:DescribeTrafficSources",
        "vpc-lattice:RegisterTargets",
        "vpc-lattice:DeregisterTargets"
    ],
    "Resource": "*"
}
]
```

- Wenn die Startvorlage für Ihre Auto-Scaling-Gruppe nicht die richtigen Einstellungen für VPC Lattice enthält, z. B. eine kompatible Sicherheitsgruppe, müssen Sie die Startvorlage aktualisieren. Bestehende Instances werden nicht mit den neuen Einstellungen aktualisiert, wenn die Startvorlage geändert wird. Um bestehende Instances zu aktualisieren, können Sie eine Instance-Aktualisierung starten, um die Instances zu ersetzen. Weitere Informationen finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).
- Bevor Sie die VPC Lattice-Integritätsprüfungen für Ihre Auto-Scaling-Gruppe aktivieren, können Sie eine anwendungsbasierte Zustandsprüfung konfigurieren, um sicherzustellen, dass Ihre Anwendung wie erwartet reagiert. Weitere Informationen finden Sie unter [Zustandsprüfungen für Ihre Zielgruppen](#) im VPC Lattice Benutzerhandbuch.

Sicherheitsgruppen: Eingehende und ausgehende Regeln

Sicherheitsgruppen fungieren als Firewall für zugehörige EC2-Instances und kontrollieren sowohl den eingehenden als auch den ausgehenden Datenverkehr auf Instance-Ebene.

Note

Die Netzwerkkonfiguration ist so komplex, dass wir Ihnen dringend empfehlen, eine neue Sicherheitsgruppe für die Verwendung mit VPC Lattice zu erstellen. Es macht es auch einfacher Support , Ihnen zu helfen, falls Sie Kontakt mit ihnen aufnehmen müssen. Die folgenden Abschnitte basieren auf der Annahme, dass Sie dieser Empfehlung folgen. Weitere Informationen zum Erstellen von Sicherheitsgruppen für VPC Lattice, die Sie mit Ihrer Auto-Scaling-Gruppe verwenden können, finden Sie unter [Steuern des Datenverkehrs mithilfe von Sicherheitsgruppen](#) im VPC Lattice Benutzerhandbuch. Weitere Informationen zur Behebung von Problemen mit dem Datenfluss finden Sie im VPC Lattice Benutzerhandbuch.

Informationen zum Erstellen einer Sicherheitsgruppe finden Sie unter [Erstellen einer Sicherheitsgruppe](#) im Amazon EC2 EC2-Benutzerhandbuch. Verwenden Sie die folgende Tabelle, um zu bestimmen, welche Optionen Sie auswählen müssen.

Option	Wert	
Name	Ein Name, den Sie sich leicht merken können.	
Description	Eine Beschreibung, die Ihnen hilft, die Sicherheitsgruppe zu identifizieren.	
VPC	Dieselbe VPC wie die Auto-Scaling-Gruppe	

Regeln für eingehenden Datenverkehr

Wenn Sie eine Sicherheitsgruppe erstellen, verfügt sie über keine Regeln für den eingehenden Datenverkehr. Es ist kein eingehender Verkehr von Clients innerhalb eines VPC-Lattice-Servicenetzes zu Ihrer Instance erlaubt, bis Sie der Sicherheitsgruppe Regeln für eingehenden Verkehr hinzufügen.

Damit Clients innerhalb eines VPC Lattice-Dienstnetzes eine Verbindung zu Instances in Ihrer Auto-Scaling-Gruppe herstellen können, muss die Sicherheitsgruppe für Ihre Auto Scaling-Gruppe korrekt eingerichtet sein. Geben Sie ihm in diesem Fall eine Regel für eingehenden Datenverkehr, um Datenverkehr über den Namen der AWS verwalteten Präfixliste für VPC Lattice statt über eine bestimmte IP-Adresse zuzulassen. Die VPC Lattice-Präfixliste ist ein Bereich von IP-Adressen, die von VPC Lattice in CIDR-Notation verwendet werden. Weitere Informationen finden Sie unter [Arbeiten mit AWS verwalteten Präfixlisten](#) im Amazon VPC-Benutzerhandbuch.

Informationen zum Hinzufügen von Regeln zu einer Sicherheitsgruppe finden Sie unter [Sicherheitsgruppenregeln konfigurieren](#) im Amazon VPC-Benutzerhandbuch. Verwenden Sie die folgende Tabelle, um zu bestimmen, welche Optionen Sie auswählen sollten.

Option	Wert	
HTTP-Regel	Typ: HTTP	

Option	Wert
	Quelle: com.amazonaws. <i>region</i> .vpc-lattice
HTTPS-Regel	Typ: HTTPS Quelle: com.amazonaws. <i>region</i> .vpc-lattice

Die Sicherheitsgruppe ist zustandsbehaftet: Sie lässt den Datenverkehr von Clients innerhalb des VPC Lattice Service-Netzwerks zu Instances in Ihrer Auto-Scaling-Gruppe zu und sendet dann die Antwort an den Client zurück, den sie zuvor verlassen hat.

Regeln für ausgehenden Datenverkehr

Standardmäßig enthält eine Sicherheitsgruppe eine ausgehende Regel, die den gesamten ausgehenden Datenverkehr zulässt. Sie können diese Standardregel optional entfernen und eine Regel für ausgehenden Datenverkehr hinzufügen, um bestimmten Sicherheitsanforderungen gerecht zu werden.

Einschränkungen

- [Gruppen mit gemischten Instanzen werden unterstützt](#). Wenn Sie eine VPC Lattice-Zielgruppe einer Auto Scaling Scaling-Gruppe zuordnen, die über eine Richtlinie für gemischte Instanzen verfügt, verteilt der Load Balancing-Algorithmus die Last gleichmäßig auf alle verfügbaren Ressourcen und geht davon aus, dass die Instances ähnlich genug sind, um gleiche Lasten zu bewältigen.

Hinzufügen einer VPC-Lattice-Zielgruppe zu Ihrer Auto-Scaling-Gruppe

In diesem Thema wird beschrieben, wie Sie eine VPC Lattice-Zielgruppe an eine Auto-Scaling-Gruppe anhängen. Außerdem wird beschrieben, wie VPC Lattice-Zustandsprüfungen aktiviert werden, damit Amazon EC2 Auto Scaling Instances ersetzen kann, die VPC Lattice als fehlerhaft meldet.

Standardmäßig ersetzt Amazon EC2 Auto Scaling nur Instances, die aufgrund von Amazon EC2-Zustandsprüfungen nicht zustandsbehaftet oder nicht erreichbar sind. Wenn Sie die VPC Lattice-

Zustandsprüfungen aktivieren, kann Amazon EC2 Auto Scaling eine laufende Instance ersetzen, wenn eine der VPC Lattice-Zielgruppen, die Sie der Auto Scaling-Gruppe zuordnen, diese als fehlerhaft meldet. Weitere Informationen finden Sie unter [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#).

 **Important**

Bevor Sie fortfahren, müssen Sie alle im vorherigen Abschnitt genannten [Voraussetzungen](#) erfüllen.

Fügen Sie eine VPC-Lattice-Zielgruppe hinzu

Sie können einer Auto Scaling Scoping-Gruppe eine oder mehrere Zielgruppen zuordnen, wenn Sie die Gruppe erstellen oder aktualisieren.

Console

Befolgen Sie die Schritte in diesem Abschnitt, um die Konsole für Folgendes zu verwenden:

- Anhängen einer VPC Lattice-Zielgruppe an eine Auto-Scaling-Gruppe
- Aktivieren der Zustandsprüfungen für VPC Lattice

So fügen Sie eine VPC Lattice-Zielgruppe einer neuen Auto Scaling-Gruppe hinzu

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Wählen Sie in der Navigationsleiste oben die AWS-Region aus, in der Sie Ihre Gruppe erstellt haben.
3. Wählen Sie Erstellen einer Auto-Scaling-Gruppe aus.
4. Wählen Sie in den Schritten 1 und 2 die gewünschten Optionen aus und fahren Sie mit Schritt 3: Konfigurieren von erweiterten Optionen fort.
5. Wählen Sie für VPC Lattice-Integrationsoptionen die Option An VPC Lattice-Dienst anhängen.
6. Wählen Sie unter VPC Lattice-Zielgruppe auswählen Ihre Zielgruppe aus.
7. (Optional) Wählen Sie für Zustandsprüfungen und Zusätzliche Zustandsprüfungstypen die Option VPC Lattice-Zustandsprüfungen aktivieren aus.

8. (Optional) Geben Sie unter Karenzzeit für die Zustandsprüfung die Zeit in Sekunden ein. Diese Zeitspanne gibt an, wie lange Amazon EC2 Auto Scaling warten muss, bevor der Integritätsstatus einer Instance überprüft wird, nachdem sie den InService Status erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).
9. Fahren Sie mit dem Erstellen der Auto-Scaling-Gruppe fort. Ihre Instances werden automatisch in der VPC Lattice-Zielgruppe registriert, nachdem die Auto-Scaling-Gruppe erstellt wurde.

So fügen Sie eine VPC Lattice-Zielgruppe einer bestehenden Auto Scaling-Gruppe hinzu

Gehen Sie wie folgt vor, um eine Zielgruppe für einen Dienst an eine vorhandene Auto-Scaling-Gruppe anzuhängen.

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer Auto-Scaling-Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option VPC Lattice-Integrationsoptionen und Bearbeiten aus.
4. Wählen Sie für VPC Lattice-Integrationsoptionen die Option An VPC Lattice-Dienst anhängen aus.
5. Wählen Sie unter VPC Lattice-Zielgruppe auswählen Ihre Zielgruppe aus.
6. Wählen Sie Aktualisieren aus.

Wenn Sie mit dem Hinzufügen der Zielgruppe fertig sind, können Sie optional die Zustandsprüfungen aktivieren, die diese Zielgruppe verwenden.

So aktivieren Sie die VPC Lattice Zustandsprüfungen

1. Wählen Sie auf der Registerkarte Details die Option Zustandsprüfungen, Bearbeiten aus.
2. Wählen Sie für Zustandsprüfungen und Zusätzliche Zustandsprüfungstypen die Option VPC Lattice-Zustandsprüfungen aktivieren aus.
3. Geben Sie unter Frist für Zustandsprüfungen die Zeit in Sekunden ein. Diese Zeitspanne gibt an, wie lange Amazon EC2 Auto Scaling warten muss, bevor der Integritätsstatus

einer Instance überprüft wird, nachdem sie den InService Status erreicht hat. Weitere Informationen finden Sie unter [Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest](#).

4. Wählen Sie Aktualisieren aus.

AWS CLI

Folgen Sie den Schritten in diesem Abschnitt, um Folgendes AWS CLI zu verwenden:

- Anhängen einer VPC Lattice-Zielgruppe an eine Auto-Scaling-Gruppe
- Aktivieren der Zustandsprüfungen für VPC Lattice

So fügen Sie eine VPC Lattice-Zielgruppe einer Auto Scaling-Gruppe zu

Verwenden Sie den folgenden [create-auto-scaling-group](#) Befehl, um eine Auto Scaling Scaling-Gruppe zu erstellen und gleichzeitig eine VPC Lattice-Zielgruppe anzuhängen, indem Sie ihren Amazon Resource Name (ARN) angeben.

Ersetzen Sie die Beispielwerte für `--auto-scaling-group-name`, `--vpc-zone-identifizier`, `--min-size` und `--max-size`. Für die Option `--launch-template` ersetzen Sie `my-launch-template` und `1` durch den Namen und die Version der Startvorlage, die Sie für Instances erstellt haben, die in einer VPC Lattice-Zielgruppe registriert sind. Bei der Option `--traffic-sources` ersetzen Sie den Beispiel-ARN durch den ARN Ihrer VPC Lattice-Zielgruppe.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \  
  --vpc-zone-identifizier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
  --min-size 1 --max-size 5 \  
  --traffic-sources "Identifizier=arn:aws:vpc-lattice:region:account-id:targetgroup/tg-0e2f2665eEXAMPLE"
```

Verwenden Sie den folgenden [attach-traffic-sources](#) Befehl, um eine VPC Lattice-Zielgruppe an eine Auto Scaling Scaling-Gruppe anzuhängen, nachdem sie bereits erstellt wurde.

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \  
  --traffic-sources "Identifizier=arn:aws:vpc-lattice:region:account-id:targetgroup/tg-0e2f2665eEXAMPLE"
```

So aktivieren Sie die Zustandsprüfungen für VPC Lattice

Wenn Sie eine anwendungsbasierte Zustandsprüfung für Ihre VPC Lattice-Zielgruppe konfiguriert haben, können Sie diese Zustandsprüfung aktivieren. Verwenden Sie den [update-auto-scaling-group](#) Befehl [create-auto-scaling-group](#) oder mit der `--health-check-type` Option und dem Wert `VPC_LATTICE` Um den Kulanzzeitraum für die von Ihrer Auto-Scaling-Gruppe durchgeführten Integritätsprüfungen anzugeben, schließen Sie die Option `--health-check-grace-period` ein und geben Sie ihren Wert in Sekunden an.

```
--health-check-type "VPC_LATTICE" --health-check-grace-period 60
```

Eine VPC-Lattice-Zielgruppe abtrennen

Wenn Sie VPC Lattice nicht mehr benötigen, gehen Sie wie folgt vor, um die Zielgruppe von Ihrer Auto-Scaling-Gruppe zu trennen.

Console

Befolgen Sie die Schritte in diesem Abschnitt, um die Konsole für Folgendes zu verwenden:

- Trennen einer VPC Lattice-Zielgruppe von einer Auto-Scaling-Gruppe
- Schalten Sie die Zustandsprüfungen für VPC Lattice aus

So trennen Sie eine VPC Lattice-Zielgruppe von einer Auto-Scaling-Gruppe

1. Öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/> und wählen Sie im Navigationsbereich Auto Scaling Groups aus.
2. Aktivieren Sie das Kontrollkästchen neben einer vorhandenen Gruppe.

Im unteren Teil der Seite wird ein geteilter Bereich geöffnet.

3. Wählen Sie auf der Registerkarte Details die Option VPC Lattice-Integrationsoptionen und Bearbeiten aus.
4. Wählen Sie unter VPC Lattice Integration Options das Löschsymbol (X) neben der Zielgruppe aus.
5. Wählen Sie Aktualisieren aus.

Wenn Sie mit dem Trennen der Zielgruppe fertig sind, können Sie die VPC Lattice-Zustandsprüfungen deaktivieren.

So deaktivieren Sie die VPC Lattice Zustandsprüfungen

1. Wählen Sie auf der Registerkarte Details die Option Zustandsprüfungen, Bearbeiten aus.
2. Deaktivieren Sie für Zustandsprüfungen und Zusätzliche Zustandsprüfungstypen die Option VPC Lattice-Zustandsprüfungen aktivieren.
3. Wählen Sie Aktualisieren aus.

AWS CLI

Folgen Sie den Schritten in diesem Abschnitt, um Folgendes AWS CLI zu verwenden:

- Trennen einer VPC Lattice-Zielgruppe von einer Auto-Scaling-Gruppe
- Schalten Sie die Zustandsprüfungen für VPC Lattice aus

Verwenden Sie den [detach-traffic-sources](#) Befehl, um eine Zielgruppe von Ihrer Auto Scaling Scaling-Gruppe zu trennen, wenn Sie sie nicht mehr benötigen.

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \  
  --traffic-sources "Identifizier=arn:aws:vpc-lattice:region:account-id:targetgroup/  
tg-0e2f2665eEXAMPLE"
```

Verwenden Sie den Befehl, um die Integritätsprüfungen für eine Auto Scaling Scaling-Gruppe so zu aktualisieren, dass sie keine VPC Lattice-Integritätsprüfungen mehr verwendet. [update-auto-scaling-group](#) Geben Sie die Option `--health-check-type` und den Wert **EC2** ein.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
  --health-check-type "EC2"
```

Überprüfen Sie den Anhangsstatus Ihrer VPC Lattice-Zielgruppe

Nachdem Sie eine VPC Lattice-Zielgruppe an eine Auto-Scaling-Gruppe angehängt haben, wechselt sie bei der Registrierung der Instances in der Gruppe in den Adding-Status. Wenn alle Instances der Gruppe angemeldet sind, wird sie in den Status Added versetzt. Besteht zumindest eine angemeldete Instance die Zustandsprüfungen, wird er in den Status InService versetzt. Wenn sich die Zielgruppe im Zustand InService befindet, kann Amazon EC2 Auto Scaling alle als fehlerhaft gemeldeten Instances beenden und ersetzen. Wenn keine registrierten Instances die

Zustandsprüfungen bestehen (z. B. aufgrund einer falsch konfigurierten Zustandsprüfung), tritt die Zielgruppe nicht in den Zustand `InService` ein. Amazon EC2 Auto Scaling beendet und ersetzt die Instances nicht.

Wenn Sie eine Zielgruppe für einen Dienst abtrennen, wird sie in den Status `Removing` versetzt, solange sie die Instances der Gruppe abmeldet. Nach der Abmeldung werden die Instances weiterhin ausgeführt. Standardmäßig ist `Connection Draining` (Verzögerung der Registrierungsaufhebung) aktiviert. Ist `Connection Draining` aktiviert, wartet VPC Lattice darauf, dass aktive Anforderungen abgeschlossen werden oder das maximale Zeitlimit abgelaufen ist (je nachdem, was zuerst eintritt), bevor die Instances abgemeldet werden.

Sie können den Status des Anhangs überprüfen, indem Sie `AWS Command Line Interface (AWS CLI)` oder verwenden. `AWS SDKs` Sie können den Status des Anhangs nicht von der Konsole aus überprüfen.

AWS CLI Um den Status des Anhangs zu überprüfen

Der folgende [describe-traffic-sources](#) Befehl gibt den Anhangsstatus aller Verkehrsquellen für die angegebene Auto Scaling Scaling-Gruppe zurück.

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

Das Beispiel gibt den ARN der VPC Lattice-Zielgruppe zurück, die an die Auto-Scaling-Gruppe angehängt ist, zusammen mit dem Anhangsstatus der Zielgruppe im Element `State`.

```
{
  "TrafficSources": [
    {
      "Identifier": "arn:aws:vpc-lattice:region:account-
      id:targetgroup/tg-0e2f2665eEXAMPLE",
      "State": "InService",
      "Type": "vpc-lattice"
    }
  ]
}
```

Wird EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen verwendet

Amazon EventBridge, früher CloudWatch Events genannt, hilft Ihnen bei der Einrichtung ereignisgesteuerter Regeln, die Ressourcen überwachen und Zielaktionen einleiten, die andere AWS Dienste nutzen.

Ereignisse von Amazon EC2 Auto Scaling werden nahezu EventBridge in Echtzeit übermittelt. Sie können EventBridge Regeln einrichten, die als Reaktion auf eine Vielzahl dieser Ereignisse programmgesteuerte Aktionen und Benachrichtigungen aufrufen. Während Instances beispielsweise gerade gestartet oder beendet werden, können Sie eine AWS Lambda Funktion aufrufen, um eine vorkonfigurierte Aufgabe auszuführen.

Zu den Zielen von EventBridge Regeln können AWS Lambda Funktionen, Amazon SNS SNS-Themen, API-Ziele, Event-Busse usw. gehören. AWS-Konten Informationen zu unterstützten Zielen finden Sie unter [EventBridge Amazon-Ziele](#) im EventBridge Amazon-Benutzerhandbuch.

Erstellen Sie zunächst EventBridge Regeln anhand eines Beispiels mit einem Amazon SNS SNS-Thema und einer EventBridge Regel. Wenn ein Benutzer dann eine Aktualisierung der Instance startet, benachrichtigt Amazon SNS Sie per E-Mail, wenn ein Checkpoint erreicht wird. Weitere Informationen finden Sie unter [Erstellen Sie EventBridge Regeln für Instance-Aktualisierungsereignisse](#).

Inhalt

- [Ereignis-Referenz für Amazon EC2 Auto Scaling](#)
- [Beispiele für Ereignisse und Muster zur Instanzaktualisierung](#)
- [Beispielereignisse und -muster in einem warmen Pool](#)
- [Verwenden Sie EventBridge Amazon-Regeln, um Aktionen zu automatisieren](#)

Ereignis-Referenz für Amazon EC2 Auto Scaling

Mit Amazon können Sie Regeln erstellen EventBridge, die eingehenden Ereignissen entsprechen, und diese zur Verarbeitung an Ziele weiterleiten.

Inhalt

- [Lebenszyklus-Aktions-Ereignisse](#)

- [Erfolgreiche Skalierungsereignisse](#)
- [Fehlgeschlagene Skalierungsereignisse](#)

Lebenszyklus-Aktions-Ereignisse

Wenn Sie Ihrer Auto Scaling-Gruppe Lifecycle-Hooks hinzufügen, sendet Amazon EC2 Auto Scaling Ereignisse an den EventBridge Zeitpunkt, an dem eine Instance in einen Wartestatus übergeht. Ereignisse werden auf die bestmögliche Weise ausgegeben.

Event types (Ereignistypen)

- [Aufskalierungs-Lebenszyklus-Aktion](#)
- [Herunterskalierungs-Lebenszyklus-Aktion](#)

Aufskalierungs-Lebenszyklus-Aktion

Im folgenden Beispiereignis hat Amazon EC2 Auto Scaling eine Instance aufgrund eines Start-Lebenszyklus-Hooks in einen Pending:Wait-Status versetzt.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

Herunterskalierungs-Lebenszyklus-Aktion

Im folgenden Beispielergebnis hat Amazon EC2 Auto Scaling eine Instance aufgrund eines Beendigungs-Lebenszyklus-Hooks in einen `Terminating:Wait`-Status versetzt.

Important

Wenn eine Auto-Scaling-Gruppe die Instances beim Abskalieren an einen warmen Pool zurückgibt, kann die Rückgabe von Instances an den warmen Pool auch EC2 Instance-terminate Lifecycle Action-Ereignisse erzeugen. Ereignisse, die ausgelöst werden, wenn eine Instance beim Abskalieren in den Wartestatus wechselt, haben `WarmPool` als den Wert für `Destination`. Weitere Informationen finden Sie unter [Instance reuse policy](#).

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata": "additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

Erfolgreiche Skalierungsereignisse

Die folgenden Beispiele zeigen die Ereignistypen für erfolgreiche Skalierungsereignisse. Ereignisse werden auf die bestmögliche Weise ausgegeben.

Event types (Ereignistypen)

- [Erfolgreiche Aufskalierungsereignisse](#)
- [Erfolgreiche Abskalierungsereignisse](#)

Erfolgreiche Aufskalierungsereignisse

Im folgenden Beispielergebnis hat Amazon EC2 Auto Scaling erfolgreich eine Instance gestartet.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "InProgress",
    "Description": "Launching a new EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

Erfolgreiche Abskalierungsereignisse

Im folgenden Beispielergebnis hat Amazon EC2 Auto Scaling erfolgreich eine Instance beendet.

Important

Wenn eine Auto-Scaling-Gruppe die Instances beim Abskalieren an einen warmen Pool zurückgibt, kann die Rückgabe von Instances an den warmen Pool auch EC2 Instance Terminate Successful-Ereignisse erzeugen. Ereignisse, die ausgelöst werden, wenn eine Instance erfolgreich in den warmen Pool zurückkehrt, haben `WarmPool` als den Wert für `Destination`. Weitere Informationen finden Sie unter [Instance reuse policy](#).

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "InProgress",
    "Description": "Terminating EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    }
  },
  "RequestId": "12345678-1234-1234-1234-123456789012",
  "StatusMessage": "",
  "EndTime": "yyyy-mm-ddThh:mm:ssZ",
  "EC2InstanceId": "i-1234567890abcdef0",
  "StartTime": "yyyy-mm-ddThh:mm:ssZ",
  "Cause": "description-text",
  "Origin": "AutoScalingGroup",
  "Destination": "EC2"
}
```

```
}  
}
```

Fehlgeschlagene Skalierungsereignisse

Die folgenden Beispiele zeigen die Ereignistypen für fehlgeschlagene Skalierungsereignisse. Ereignisse werden auf die bestmögliche Weise ausgegeben.

Event types (Ereignistypen)

- [Fehlgeschlagene Aufskalierungsereignisse](#)
- [Fehlgeschlagene Abskalierungsereignisse](#)

Fehlgeschlagene Aufskalierungsereignisse

Im folgenden Beispielergebnis ist der Start einer Instance durch Amazon EC2 Auto Scaling fehlgeschlagen.

```
{  
  "version": "0",  
  "id": "12345678-1234-1234-1234-123456789012",  
  "detail-type": "EC2 Instance Launch Unsuccessful",  
  "source": "aws.autoscaling",  
  "account": "123456789012",  
  "time": "yyyy-mm-ddThh:mm:ssZ",  
  "region": "us-west-2",  
  "resources": [  
    "auto-scaling-group-arn",  
    "instance-arn"  
  ],  
  "detail": {  
    "StatusCode": "Failed",  
    "AutoScalingGroupName": "my-asg",  
    "ActivityId": "87654321-4321-4321-4321-210987654321",  
    "Details": {  
      "Availability Zone": "us-west-2b",  
      "Subnet ID": "subnet-12345678"  
    },  
    "RequestId": "12345678-1234-1234-1234-123456789012",  
    "StatusMessage": "message-text",  
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",  
    "EC2InstanceId": "i-1234567890abcdef0",  
  },  
}
```

```

    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}

```

Fehlgeschlagene Abskalierungsereignisse

Im folgenden Beispielergebnis ist die Beendigung einer Instance durch Amazon EC2 Auto Scaling fehlgeschlagen.

Important

Wenn eine Auto-Scaling-Gruppe keine Instances beim Abskalieren an einen warmen Pool zurückgibt, kann die Rückgabe von Instances an den warmen Pool auch EC2 Instance Terminate Unsuccessful-Ereignisse erzeugen. Ereignisse, die ausgelöst werden, wenn eine Instance nicht erfolgreich in den warmen Pool zurückkehrt, haben `WarmPool` als den Wert für `Destination`. Weitere Informationen finden Sie unter [Instance reuse policy](#).

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    }
  },
}

```

```
"RequestId": "12345678-1234-1234-1234-123456789012",
"StatusMessage": "message-text",
"EndTime": "yyyy-mm-ddThh:mm:ssZ",
"EC2InstanceId": "i-1234567890abcdef0",
"StartTime": "yyyy-mm-ddThh:mm:ssZ",
"Cause": "description-text",
"Origin": "AutoScalingGroup",
"Destination": "EC2"
}
}
```

Beispiele für Ereignisse und Muster zur Instanzaktualisierung

Mit Amazon können Sie Regeln erstellen EventBridge, die eingehenden Ereignissen entsprechen, und diese zur Verarbeitung an Ziele weiterleiten.

Inhalt

- [Instance-Aktualisierungsereignisse](#)
- [Die Aktualisierung der Instanz ersetzt Ereignisse im Lebenszyklus des Root-Volumes](#)

Instance-Aktualisierungsereignisse

Die folgenden Beispiele zeigen Ereignisse für das Instance-Aktualisierungs-Feature. Ereignisse werden auf die bestmögliche Weise ausgegeben.

Event types (Ereignistypen)

- [Prüfpunkt erreicht](#)
- [Instance-Aktualisierung gestartet](#)
- [Instance-Aktualisierung erfolgreich](#)
- [Instance-Aktualisierung fehlgeschlagen](#)
- [Instance-Aktualisierung abgebrochen](#)
- [Das Rollback der Instanzaktualisierung wurde gestartet](#)
- [Das Rollback der Instanzaktualisierung war erfolgreich](#)
- [Das Rollback der Instanzaktualisierung ist fehlgeschlagen](#)

Prüfpunkt erreicht

Während einer Instance-Aktualisierung sendet Amazon EC2 Auto Scaling das folgende Ereignis, wenn die Anzahl der ersetzten Instances den für den Prüfpunkt definierten prozentualen Schwellenwert erreicht.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Checkpoint Reached",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "ab00cf8f-9126-4f3c-8010-dbb8cad6fb86",
    "AutoScalingGroupName": "my-asg",
    "CheckpointPercentage": "50",
    "CheckpointDelay": "300"
  }
}
```

Instance-Aktualisierung gestartet

Amazon EC2 Auto Scaling sendet das folgende Ereignis, wenn sich der Zustand einer Instance-Aktualisierung auf InProgress ändert.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Started",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
```

```

    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}

```

Instance-Aktualisierung erfolgreich

Amazon EC2 Auto Scaling sendet das folgende Ereignis, wenn sich der Zustand einer Instance-Aktualisierung auf `Successful` ändert.

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}

```

Instance-Aktualisierung fehlgeschlagen

Amazon EC2 Auto Scaling sendet das folgende Ereignis, wenn sich der Zustand einer Instance-Aktualisierung auf `Failed` ändert.

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Failed",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ]
}

```

```

],
"detail": {
  "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
  "AutoScalingGroupName": "my-asg"
}
}

```

Instance-Aktualisierung abgebrochen

Amazon EC2 Auto Scaling sendet das folgende Ereignis, wenn sich der Zustand einer Instance-Aktualisierung auf Cancelled ändert.

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Cancelled",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}

```

Das Rollback der Instanzaktualisierung wurde gestartet

Amazon EC2 Auto Scaling sendet das folgende Ereignis, wenn sich der Zustand einer Instance-Aktualisierung auf RollbackInProgress ändert.

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Started",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",

```

```

"resources": [
  "auto-scaling-group-arn"
],
"detail": {
  "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
  "AutoScalingGroupName": "my-asg"
}
}

```

Das Rollback der Instanzaktualisierung war erfolgreich

Amazon EC2 Auto Scaling sendet das folgende Ereignis, wenn sich der Zustand einer Instance-Aktualisierung auf RollbackSuccessful ändert.

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}

```

Das Rollback der Instanzaktualisierung ist fehlgeschlagen

Amazon EC2 Auto Scaling sendet das folgende Ereignis, wenn sich der Zustand einer Instance-Aktualisierung auf Failed ändert.

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Failed",
  "source": "aws.autoscaling",
  "account": "123456789012",

```

```

"time": "yyyy-mm-ddThh:mm:ssZ",
"region": "us-west-2",
"resources": [
  "auto-scaling-group-arn"
],
"detail": {
  "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
  "AutoScalingGroupName": "my-asg"
}
}

```

Die Aktualisierung der Instanz ersetzt Ereignisse im Lebenszyklus des Root-Volumes

Wenn Sie Ihrer Auto Scaling-Gruppe Lifecycle-Hooks hinzufügen, sendet Amazon EC2 Auto Scaling Ereignisse an den EventBridge Zeitpunkt, an dem eine Instance in einen Wartestatus übergeht. Weitere Informationen finden Sie unter [Verwenden Sie Lifecycle-Hooks mit Replace Root-Volume](#).

Die folgenden Beispiele zeigen Ereignisse, wenn Ihre Auto Scaling Scaling-Gruppe Instances hat, die während einer Instance-Aktualisierung ein Root-Volume ersetzt werden. Ereignisse werden auf die bestmögliche Weise ausgegeben.

Beispiele für Ereignisse

- [Starten Sie die Lebenszyklusaktion „Root-Volume ersetzen](#)
- [Lebenszyklusaktion „Root-Volume ersetzen“ beenden](#)
- [Beispiel für Ereignismuster](#)

Starten Sie die Lebenszyklusaktion „Root-Volume ersetzen

Wenn eine Instance während der Aktualisierung der Instanz für den Austausch des Root-Volumes ausgewählt wird und ihr Status sich auf `ändertReplacingRootVolume:Wait`.

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [

```

```

    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-terminate-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata": "additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "AutoScalingGroup",
    "Action": "ReplaceRootVolume"
  }
}

```

Lebenszyklusaktion „Root-Volume ersetzen“ beenden

Wenn eine Instance den Austausch des Root-Volumens abgeschlossen hat und sich ihr Status auf `Pending:Wait` ändert.

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "AutoScalingGroup",
    "Action": "ReplaceRootVolume"
  }
}

```

Beispiel für Ereignismuster

Der vorherige Abschnitt enthält Beispielergebnisse, die von Amazon EC2 Auto Scaling während einer Instance-Aktualisierung mit der Strategie „Root-Volume ersetzen“ ausgelöst wurden. EventBridge Ereignismuster haben dieselbe Struktur wie die Ereignisse, denen sie entsprechen. Das Muster zitiert die Felder, die Sie abgleichen möchten, und liefert die Werte, nach denen Sie suchen. Die folgenden Felder des Ereignisses bilden das in der Regel definierte Ereignismuster, das eine Aktion aufruft:

```
"source": "aws.autoscaling"
```

Gibt an, dass das Ereignis aus Amazon EC2 Auto Scaling stammt.

```
"detail-type": "EC2 Instance-launch Lifecycle Action"
```

Identifiziert den Ereignistyp.

```
"Action": "ReplaceRootVolume"
```

Identifiziert, welche Aktion auf der Instanz ausgeführt wird.

Verwenden Sie das folgende Beispielergebnismuster, um alle Ereignisse zu erfassen, die mit Instances in Verbindung stehen, die sich darauf vorbereiten, ihre Root-Volumes zu ersetzen:

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ],
  "detail": {
    "Action": [ "ReplaceRootVolume" ]
  }
}
```

Verwenden Sie das folgende Beispielergebnismuster, um alle Ereignisse zu erfassen, die damit zusammenhängen, dass Instances nach Abschluss des Austauschs des Root-Volumes wieder in Betrieb genommen werden:

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "Action": [ "ReplaceRootVolume" ]
  }
}
```

}

Beispielereignisse und -muster in einem warmen Pool

Amazon EC2 Auto Scaling unterstützt mehrere vordefinierte Muster in Amazon EventBridge. Dies vereinfacht die Erstellung eines Ereignismusters. Sie wählen Feldwerte in einem Formular aus und EventBridge generiert das Muster für Sie. Derzeit unterstützt Amazon EC2 Auto Scaling keine vordefinierten Muster für Ereignisse, die von einer Auto-Scaling-Gruppe mit einem Warm-Pool ausgelöst werden. Sie müssen das Muster als JSON-Objekt eingeben. Dieser Abschnitt und das Thema [Erstellen Sie EventBridge Regeln für Ereignisse im warmen Pool](#) zeigen Ihnen, wie Sie ein Ereignismuster verwenden, um Ereignisse auszuwählen und sie an Ziele zu senden.

Um EventBridge Regeln zu erstellen, die nach Ereignissen im Zusammenhang mit warmen Pools filtern, an die Amazon EC2 Auto Scaling sendet EventBridge, fügen Sie die `Destination` Felder `Origin` und aus dem `detail` Abschnitt des Ereignisses hinzu.

Bei den Werten `Origin` und `Destination` kann es sich um Folgendes handeln:

EC2 | AutoScalingGroup | WarmPool

Inhalt

- [Beispielereignisse](#)
- [Beispiel für Ereignismuster](#)

Beispielereignisse

Wenn Sie Ihrer Auto Scaling-Gruppe Lifecycle-Hooks hinzufügen, sendet Amazon EC2 Auto Scaling Ereignisse an den EventBridge Zeitpunkt, an dem eine Instance in einen Wartestatus übergeht. Weitere Informationen finden Sie unter [Verwenden Sie Lifecycle-Hooks mit einem warmen Pool in der Auto Scaling Scaling-Gruppe](#).

Dieser Abschnitt enthält Beispiele für diese Ereignisse, wenn Ihre Auto-Scaling-Gruppe über einen warmen Pool verfügt. Ereignisse werden auf die bestmögliche Weise ausgegeben.

Note

Informationen zu Ereignissen, an die Amazon EC2 Auto Scaling EventBridge bei erfolgreicher Skalierung sendet, finden Sie unter [Erfolgreiche Skalierungsereignisse](#). Informationen zu

Ereignissen, bei denen die Skalierung nicht erfolgreich ist, finden Sie unter [Fehlgeschlagene Skalierungsereignisse](#).

Beispiele für Ereignisse

- [Aufskalierungs-Lebenszyklus-Aktion](#)
- [Herunterskalierungs-Lebenszyklus-Aktion](#)

Aufskalierungs-Lebenszyklus-Aktion

Ereignisse, die geliefert werden, wenn eine Instance beim Aufskalieren in den Wartestatus wechselt, haben EC2 Instance-launch Lifecycle Action als den Wert für detail-type. Im detail-Objekt zeigen die Werte für die Attribute Origin und Destination, woher die Instance kommt und wohin sie geht.

In diesem Beispiel für ein Abskalierungsereignis wird eine neue Instance gestartet und ihr Status wird in Warmed:Pending:Wait geändert, weil sie dem warmen Pool hinzugefügt wird. Weitere Informationen finden Sie unter [Lebenszyklusstatusübergänge für Instances in einem Warm Pool](#).

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "WarmPool"
  }
}
```

}

In diesem Beispiel für ein Aufskalierungsereignis ändert sich der Status der Instance in `Pending:Wait`, wenn sie aus dem warmen Pool zur Auto-Scaling-Gruppe hinzugefügt wird. Weitere Informationen finden Sie unter [Lebenszyklusstatusübergänge für Instances in einem Warm Pool](#).

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-19T00:35:52.359Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "19cc4d4a-e450-4d1c-b448-0de67EXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "WarmPool",
    "Destination": "AutoScalingGroup"
  }
}
```

Herunterskalierungs-Lebenszyklus-Aktion

Ereignisse, die geliefert werden, wenn eine Instance beim Abskalieren in den Wartestatus wechselt, haben `EC2 Instance-terminate Lifecycle Action` als den Wert für `detail-type`. Im `detail`-Objekt zeigen die Werte für die Attribute `Origin` und `Destination`, woher die Instance kommt und wohin sie geht.

In diesem Beispiel für ein Abskalierungsereignis ändert sich der Status einer Instance in `Warmed:Pending:Wait`, wenn sie an den warmen Pool zurückgegeben wird. Weitere Informationen finden Sie unter [Lebenszyklusstatusübergänge für Instances in einem Warm Pool](#).

{

```

"version": "0",
"id": "12345678-1234-1234-1234-123456789012",
"detail-type": "EC2 Instance-terminate Lifecycle Action",
"source": "aws.autoscaling",
"account": "123456789012",
"time": "2022-03-28T00:12:37.214Z",
"region": "us-west-2",
"resources": [
  "auto-scaling-group-arn"
],
"detail": {
  "LifecycleActionToken": "42694b3d-4b70-6a62-8523-09a1eEXAMPLE",
  "AutoScalingGroupName": "my-asg",
  "LifecycleHookName": "my-termination-lifecycle-hook",
  "EC2InstanceId": "i-1234567890abcdef0",
  "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
  "NotificationMetadata": "additional-info",
  "Origin": "AutoScalingGroup",
  "Destination": "WarmPool"
}
}

```

Beispiel für Ereignismuster

Im vorstehenden Abschnitt werden Beispiel-Ereignisse aufgeführt, die von Amazon EC2 Auto Scaling ausgegeben werden.

EventBridge Ereignismuster haben dieselbe Struktur wie die Ereignisse, denen sie entsprechen. Das Muster zitiert die Felder, die Sie abgleichen möchten, und liefert die Werte, nach denen Sie suchen.

Die folgenden Felder des Ereignisses bilden das in der Regel definierte Ereignismuster, das eine Aktion aufruft:

"source": "aws.autoscaling"

Gibt an, dass das Ereignis aus Amazon EC2 Auto Scaling stammt.

"detail-type": "EC2 Instance-launch Lifecycle Action"

Identifiziert den Ereignistyp.

"Origin": "EC2"

Gibt an, woher die Instance kommt.

"Destination": "*WarmPool*"

Gibt an, wohin die Instance geht.

Verwenden Sie das folgende Beispiel-Ereignismuster, um alle EC2 Instance-launch Lifecycle Action-Ereignisse zu erfassen, die Instances zugeordnet sind, die in den warmen Pool gelangen.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "Origin": [ "EC2" ],
    "Destination": [ "WarmPool" ]
  }
}
```

Verwenden Sie das folgende Beispiel-Ereignismuster, um alle EC2 Instance-launch Lifecycle Action-Ereignisse zu erfassen, die mit Instances verbunden sind, die den warmen Pool aufgrund eines Aufskalierungsereignisses verlassen.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "Origin": [ "WarmPool" ],
    "Destination": [ "AutoScalingGroup" ]
  }
}
```

Verwenden Sie das folgende Beispiel-Ereignismuster, um alle EC2 Instance-launch Lifecycle Action-Ereignisse zu erfassen, die mit Instances verknüpft sind, die direkt in der Auto-Scaling-Gruppe gestartet werden.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "Origin": [ "EC2" ],
    "Destination": [ "AutoScalingGroup" ]
  }
}
```

```
}  
}
```

Verwenden Sie das folgende Beispiel-Ereignismuster, um alle EC2 Instance-terminate Lifecycle Action-Ereignisse zu erfassen, die Instances zugeordnet sind, die beim Abskalieren in den warmen Pool zurückkehren.

```
{  
  "source": [ "aws.autoscaling" ],  
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ],  
  "detail": {  
    "Origin": [ "AutoScalingGroup" ],  
    "Destination": [ "WarmPool" ]  
  }  
}
```

Verwenden Sie das folgende Beispiel-Ereignismuster, um alle Ereignisse zu erfassen, die mit EC2 Instance-launch Lifecycle Action assoziiert sind, unabhängig vom Ursprung oder Ziel.

```
{  
  "source": [ "aws.autoscaling" ],  
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]  
}
```

Verwenden Sie EventBridge Amazon-Regeln, um Aktionen zu automatisieren

Wenn ein Ereignis von Amazon EC2 Auto Scaling ausgelöst wird, wird eine Ereignisbenachrichtigung EventBridge als JSON-Datei an Amazon gesendet. Sie können eine EventBridge Regel schreiben, um zu automatisieren, welche Aktionen ergriffen werden, wenn ein Ereignismuster der Regel entspricht. Wenn ein Ereignismuster EventBridge erkannt wird, das einem in einer Regel definierten Muster entspricht, EventBridge ruft es das in der Regel angegebene Ziel (oder die Ziele) auf.

Sie können die Beispielprozeduren in diesem Abschnitt als Ausgangspunkt verwenden.

Sie könnten auch die folgende Dokumentation nützlich finden.

- Informationen zum Ausführen benutzerdefinierter Aktionen für Instances beim Starten oder bevor sie mit einer Lambda-Funktion beendet werden, finden Sie unter [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#).

- Informationen zum Aufrufen einer Lambda-Funktion bei API-Aufrufen, die mit protokolliert CloudTrail wurden, finden Sie unter [Tutorial: AWS API-Aufrufe protokollieren EventBridge](#) im EventBridge Amazon-Benutzerhandbuch.
- Weitere Informationen zum Erstellen von Ereignisregeln finden Sie im [EventBridge Amazon-Benutzerhandbuch unter Erstellen von EventBridge Amazon-Regeln, die auf Ereignisse reagieren](#).

Topics

- [Erstellen Sie EventBridge Regeln für Instance-Aktualisierungsereignisse](#)
- [Erstellen Sie EventBridge Regeln für Ereignisse im warmen Pool](#)

Erstellen Sie EventBridge Regeln für Instance-Aktualisierungsereignisse

Im folgenden Beispiel wird eine EventBridge Regel zum Senden einer E-Mail-Benachrichtigung erstellt. Dies geschieht jedes Mal, wenn Ihre Auto-Scaling-Gruppe ein Ereignis auslöst, wenn während einer Instance-Aktualisierung ein Checkpoint erreicht wird. Das Verfahren zum Einrichten von E-Mail-Benachrichtigungen über Amazon SNS ist enthalten. Damit Sie Amazon SNS zum Versenden von E-Mail-Benachrichtigungen verwenden können, müssen Sie zunächst ein Thema erstellen und es mit Ihren E-Mail-Adressen abonnieren.

Weitere Informationen über die Instance-Aktualisierungsfunktion finden Sie unter [Verwenden Sie eine Instanzaktualisierung, um Instances in einer Auto Scaling Scaling-Gruppe zu aktualisieren](#).

Erstellen Sie ein Amazon SNS-Thema.

Ein SNS-Thema ist ein logischer Zugriffspunkt, ein Kommunikationskanal der Auto-Scaling-Gruppe zum Versenden von Benachrichtigungen. Sie erstellen ein Thema, indem Sie einen Namen dafür angeben.

Themen-Namen müssen die folgenden Anforderungen erfüllen:

- 1-256 Zeichen enthalten
- Er muss ASCII-Buchstaben mit Groß- und Kleinschreibung, Zahlen, Unterstriche oder Bindestriche enthalten.

Weitere Informationen finden Sie unter [Amazon SNS-Thema anlegen](#) im Amazon Simple Notification Service-Entwicklerhandbuch.

Amazon SNS-Thema abonnieren

Zum Empfangen der Benachrichtigungen, die die Auto-Scaling-Gruppe an das Thema sendet, müssen Sie einen Endpunkt für das Thema abonnieren. In diesem Verfahren, für Endpoint, geben Sie die E-Mailadresse an, unter der Sie die Benachrichtigungen von Amazon EC2 Auto Scaling erhalten möchten.

Weitere Informationen finden Sie unter [Amazon SNS-Thema abonnieren](#) im Amazon Simple Notification Service-Entwicklerhandbuch.

Bestätigen Ihres Amazon SNS-Abonnements

Amazon SNS sendet eine Bestätigungs-E-Mail an die E-Mail-Adresse, die Sie im vorherigen Schritt angegeben haben.

Stellen Sie sicher, dass Sie die E-Mail unter AWS Benachrichtigungen öffnen und den Link zur Bestätigung des Abonnements auswählen, bevor Sie mit dem nächsten Schritt fortfahren.

Sie erhalten eine Bestätigungsnachricht von. AWS Amazon SNS ist jetzt so konfiguriert, dass Benachrichtigungen empfangen und als E-Mail an die angegebene E-Mail-Adresse gesendet werden.

Weiterleiten von Ereignissen an Ihr Amazon SNS-Thema

Erstellen Sie eine Regel, die den ausgewählten Ereignissen entspricht, und leiten Sie sie an Ihr Amazon SNS-Thema weiter, um abonnierte E-Mail-Adressen zu benachrichtigen.

So erstellen Sie eine Regel, die Benachrichtigungen an Ihr Amazon-SNS-Thema sendet

1. Öffnen Sie die EventBridge Amazon-Konsole unter <https://console.aws.amazon.com/events/>.
2. Wählen Sie im Navigationsbereich Regeln aus.
3. Wählen Sie Regel erstellen aus.
4. Zum Define rule detail (Festlegen der Regeldetails) gehen Sie folgendermaßen vor:

- a. Geben Sie für die Regel einen Name (Namen) und optional eine Beschreibung ein.

Eine Regel darf nicht denselben Namen wie eine andere Regel in derselben Region und auf demselben Event Bus haben.

- b. Bei Event bus (Ereignisbus) wählen Sie default (Standard) aus. Wenn ein AWS Service in Ihrem Konto ein Ereignis generiert, wird es immer an den Standard-Event-Bus Ihres Kontos weitergeleitet.

- c. Bei Regeltyp wählen Sie Regel mit einem Ereignismuster aus.
 - d. Wählen Sie Weiter aus.
5. Bei Build event pattern (Ereignis-Muster erstellen) gehen Sie wie folgt vor:
- a. Wählen Sie als Eventquelle AWS Events oder EventBridge Partnerevents aus.
 - b. Bei Build event pattern (Ereignis-Muster erstellen) gehen Sie wie folgt vor:
 - i. Wählen Sie für Ereignisquelle die Option AWS-Services aus.
 - ii. Für AWS-Service wählen Sie Auto Scaling aus.
 - iii. Wählen Sie für Event type (Ereignistyp) die Option Instance Refresh (Instance-Aktualisierung) aus.
 - iv. Standardmäßig entspricht die Regel jedem Instance-Aktualisierungsereignis. Um eine Regel zu erstellen, die Sie benachrichtigt, wenn während einer Instance-Aktualisierung ein Checkpoint erreicht wird, wählen Sie Specific instance event(s) (Spezifische Instance-Ereignisse) und dann Auto-Scaling-Instance-Aktualisierung für EC2 aus.
 - v. Standardmäßig stimmt die Regel mit jeder Auto-Scaling-Gruppe in der Region überein. Damit die Regel mit einer bestimmten Auto-Scaling-Gruppe übereinstimmt, wählen Sie Specific group name(s) und dann eine oder mehrere Auto-Scaling-Gruppen aus.
 - vi. Wählen Sie Weiter aus.
6. Bei Select target(s) (Ziel(e) auswählen) gehen Sie wie folgt vor:
- a. Für Target types (Zieltypen), wählen Sie AWS-Service aus.
 - b. Für Select a target (Wählen Sie ein Ziel aus), wählen Sie SNS-Thema aus.
 - c. Für Topic (Thema), wählen Sie Ihr Amazon-SNS-Thema aus.
 - d. (Optional) Unter Additional settings (Zusätzliche Einstellungen) können Sie optional zusätzliche Einstellungen konfigurieren. Weitere Informationen finden Sie im [EventBridge Amazon-Benutzerhandbuch unter Erstellen von EventBridge Amazon-Regeln, die auf Ereignisse reagieren](#).
 - e. Wählen Sie Weiter aus.
7. (Optional) Bei Tags können Sie Ihrer Regel optional einen Tag oder mehrere Tags hinzufügen und dann Next (Weiter) auswählen.
8. Für Review and create (Überprüfen und erstellen), überprüfen Sie die Details der Regel und ändern Sie sie nach Bedarf. Wählen Sie dann Create rule (Regel erstellen).

Erstellen Sie EventBridge Regeln für Ereignisse im warmen Pool

Im folgenden Beispiel wird eine EventBridge Regel zum Aufrufen programmatischer Aktionen erstellt. Dies geschieht jedes Mal, wenn Ihre Auto-Scaling-Gruppe ein Ereignis ausgibt, wenn eine neue Instance zum Warm-Pool hinzugefügt wird.

Bevor Sie die Regel erstellen, erstellen Sie die AWS Lambda Funktion, die die Regel als Ziel verwenden soll. Sie müssen diese Funktion als Ziel für die Regel angeben. Das folgende Verfahren enthält nur die Schritte zum Erstellen der EventBridge Regel, die wirksam wird, wenn neue Instanzen in den warmen Pool gelangen. Ein einführendes Tutorial, das Ihnen zeigt, wie Sie eine einfache Lambda-Funktion erstellen, die aufgerufen wird, wenn ein eingehendes Ereignis einer Regel entspricht, finden Sie unter [Tutorial: Konfigurieren eines Lebenszyklus-Hook, der eine Lambda-Funktion aufruft](#).

Weitere Informationen zum Erstellen und Arbeiten mit Warm-Pools finden Sie unter [Reduzieren Sie die Latenz für Anwendungen mit langen Startzeiten mithilfe warmer Pools](#).

So erstellen Sie eine Ereignisregel, die eine Lambda-Funktion aufruft

1. Öffnen Sie die EventBridge Amazon-Konsole unter <https://console.aws.amazon.com/events/>.
2. Wählen Sie im Navigationsbereich Regeln aus.
3. Wählen Sie Regel erstellen aus.
4. Zum Define rule detail (Festlegen der Regeldetails) gehen Sie folgendermaßen vor:
 - a. Geben Sie für die Regel einen Name (Namen) und optional eine Beschreibung ein.

Eine Regel darf nicht denselben Namen wie eine andere Regel in derselben Region und auf demselben Event Bus haben.

- b. Bei Event bus (Ereignisbus) wählen Sie default (Standard) aus. Wenn ein AWS-Service in Ihrem Konto ein Ereignis generiert, wird es immer an den Standard-Event-Bus Ihres Kontos weitergeleitet.
 - c. Bei Regeltyp wählen Sie Regel mit einem Ereignismuster aus.
 - d. Wählen Sie Weiter aus.
 5. Bei Build event pattern (Ereignis-Muster erstellen) gehen Sie wie folgt vor:
 - a. Wählen Sie als Eventquelle AWS Events oder EventBridge Partnerevents aus.

- b. Wählen Sie für Ereignismuster die Option Benutzerdefiniertes Muster (JSON-Editor) und fügen Sie das folgende Muster in das Feld Ereignismuster ein, *italics* wobei Sie den Text durch den Namen Ihrer Auto Scaling Scaling-Gruppe ersetzen.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "AutoScalingGroupName": [ "my-asg" ],
    "Origin": [ "EC2" ],
    "Destination": [ "WarmPool" ]
  }
}
```

Um eine Regel zu erstellen, die mit anderen Ereignissen übereinstimmt, ändern Sie das Ereignismuster. Weitere Informationen finden Sie unter [Beispiel für Ereignismuster](#).

- c. Wählen Sie Weiter aus.
6. Bei Select target(s) (Ziel(e) auswählen) gehen Sie wie folgt vor:
 - a. Für Target types (Zieltypen), wählen Sie AWS-Service aus.
 - b. Für Select a target (Ein Ziel auswählen), wählen Sie Lambda function (Lambda-Funktion) aus.
 - c. Für Function (Funktion) wählen Sie die Funktion aus, an die Sie die Ereignisse senden möchten.
 - d. (Optional) Für Configure version/alias (Version/Alias konfigurieren), geben Sie Versions- und Aliaseinstellungen für die Ziel-Lambda-Funktion ein.
 - e. (Optional) Für Additional settings (Zusätzliche Einstellungen), geben Sie je nach Bedarf zusätzliche Einstellungen für Ihre Anwendung ein. Weitere Informationen finden Sie im [EventBridge Amazon-Benutzerhandbuch unter Erstellen von EventBridge Amazon-Regeln, die auf Ereignisse reagieren](#).
 - f. Wählen Sie Weiter aus.
 7. (Optional) Bei Tags können Sie Ihrer Regel optional einen Tag oder mehrere Tags hinzufügen und dann Next (Weiter) auswählen.
 8. Für Review and create (Überprüfen und erstellen), überprüfen Sie die Details der Regel und ändern Sie sie nach Bedarf. Wählen Sie dann Create rule (Regel erstellen).

Stellen Sie Netzwerkkonnektivität für Ihre Auto-Scaling-Instances mit Amazon VPC bereit

Amazon Virtual Private Cloud (Amazon VPC) ist ein Service, mit dem Sie AWS Ressourcen wie Auto Scaling Scaling-Gruppen in einem logisch isolierten virtuellen Netzwerk starten können, das Sie definieren.

Ein Subnetz in einer Amazon VPC ist eine Unterteilung innerhalb einer Availability Zone, die durch ein Segment des IP-Adressbereichs der VPC definiert wird. Mithilfe von Subnetzen können Sie Instances auf Grundlage Ihrer Sicherheits- und Betriebsanforderungen gruppieren. Subnetze befinden sich vollständig innerhalb der Availability Zone, in der sie erstellt wurden. Sie starten Auto-Scaling-Instances innerhalb der Subnetze.

Sie müssen ein Internet-Gateway erstellen und Ihrem VPC hinzufügen, um die Kommunikation zwischen dem Internet und den Instances in den Subnetzen zu aktivieren. Ein Internet-Gateway ermöglicht den Ressourcen innerhalb der Subnetze das Herstellen einer Verbindung mit dem Internet durch den Edge des Amazon EC2-Netzwerks. Wird der Datenverkehr eines Subnetzes an ein Internet-Gateway weitergeleitet, wird das Subnetz als öffentliches Subnetz bezeichnet. Wird der Datenverkehr eines Subnetzes nicht an ein Internet-Gateway weitergeleitet, wird das Subnetz als privates Subnetz bezeichnet. Verwenden Sie öffentliche Subnetze für Ressourcen, die mit dem Internet verbunden sein müssen, und private Subnetze für Ressourcen, die nicht mit dem Internet verbunden sein müssen. Weitere Informationen zur Bereitstellung von Internetzugriff auf Instances in einer VPC finden Sie unter [Zugriff auf das Internet](#) im Amazon VPC-Benutzerhandbuch.

Inhalt

- [Standard-VPC](#)
- [Nicht standardmäßige VPC](#)
- [Überlegungen bei der Auswahl von VPC-Subnetzen](#)
- [IP-Adressierung in einer VPC](#)
- [Netzwerkschnittstellen in einer VPC](#)
- [Tenancy zur Instance-Platzierung](#)
- [AWS Outposts](#)
- [Weitere Ressourcen, um mehr zu erfahren über VPCs](#)

Standard-VPC

Wenn Sie Ihre Auto Scaling-Gruppe AWS-Konto nach dem 4. Dezember 2013 erstellt haben oder wenn Sie Ihre Auto Scaling Scoping-Gruppe in einer neuen erstellen AWS-Region, erstellen wir eine Standard-VPC für Sie. Diese Standard-VPC verfügt über jeweils ein Standard-Subnetz pro Availability Zone. Verfügen Sie über eine Standard-VPC, wird die Auto-Scaling-Gruppe standardmäßig in der Standard-VPC erstellt.

Sie können Ihre VPCs auf der [VPCs Seite „Ihre“](#) der Amazon VPC-Konsole einsehen.

Weitere Informationen zur Standard-VPC finden Sie unter [Standard VPCs](#) im Amazon VPC-Benutzerhandbuch.

Nicht standardmäßige VPC

Sie können weitere erstellen, VPCs indem Sie die [VPC-Dashboard-Seite im aufrufen AWS-Managementkonsole und VPC](#) erstellen auswählen.

Weitere Informationen finden Sie im [Amazon VPC-Benutzerhandbuch](#).

Note

Eine VPC umfasst alle Availability Zones in ihrer AWS-Region. Wenn Sie Ihrer VPC-Subnetze hinzufügen, wählen Sie mehrere Availability Zones aus, um sicherzustellen, dass die in diesen Subnetzen gehosteten Anwendungen hochverfügbar sind. Eine Availability Zone ist eines oder mehrere diskrete Rechenzentren mit redundanter Stromversorgung, Vernetzung und Konnektivität in einem AWS-Region. Availability Zones helfen Ihnen dabei, Produktionsanwendungen hochverfügbar, fehlertolerant und skalierbar zu machen.

Überlegungen bei der Auswahl von VPC-Subnetzen

Beachten Sie die folgenden Überlegungen bei der Auswahl von VPC-Subnetzen für Ihre Auto-Scaling-Gruppe:

- Wenn Sie Ihrer Auto-Scaling-Gruppe einen Elastic Load Balancing-Load Balancer zuordnen, können die Instances entweder in öffentlichen oder privaten Subnetzen gestartet werden. Der Load Balancer muss jedoch in öffentlichen Subnetzen erstellt werden, um die DNS-Auflösung zu unterstützen.

- Wenn Sie direkt über SSH auf Ihre Auto-Scaling-Instances zugreifen, können die Instances nur in öffentlichen Subnetzen gestartet werden.
- Wenn Sie mit AWS Systems Manager Session Manager auf Auto Scaling Scaling-Instances ohne Zugriff zugreifen, können die Instances entweder in öffentlichen oder privaten Subnetzen gestartet werden.
- Wenn Sie private Subnetze verwenden, können Sie den Auto-Scaling-Instances erlauben, über ein öffentliches NAT-Gateway auf das Internet zuzugreifen.
- Standardmäßig sind die Standardsubnetze in einer Standard-VPC öffentliche Subnetze.

IP-Adressierung in einer VPC

Beim Starten von Auto-Scaling-Instances in einer VPC wird den Instances automatisch eine private IP-Adresse aus dem CIDR-Bereich des Subnetzes, in dem die Instance gestartet ist, zugewiesen. Dies ermöglicht den Instances die Kommunikation mit anderen Instances in der VPC.

Sie können eine Startvorlage oder eine Startkonfiguration konfigurieren, um Ihren Instances öffentliche IPv4 Adressen zuzuweisen. Wenn Sie Ihren Instances öffentliche IP-Adressen zuweisen, können diese mit dem Internet oder anderen AWS Diensten kommunizieren.

Wenn Sie Instances in einem Subnetz starten, das für die automatische Zuweisung von IPv6 Adressen konfiguriert ist, erhalten sie beide IPv4 Adressen. IPv6 Andernfalls erhalten sie nur IPv4 Adressen. Weitere Informationen finden Sie unter [IPv6Adressen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zum Angeben von CIDR-Bereichen für Ihre VPC oder ein Subnetz finden Sie im [Amazon VPC-Benutzerhandbuch](#).

Amazon EC2 Auto Scaling kann beim Start der Instance automatisch zusätzliche private IP-Adressen zuweisen, wenn Sie eine Startvorlage verwenden, die zusätzliche Netzwerkschnittstellen angibt. Für jede Netzwerkschnittstelle wird eine einzelne private IP-Adresse aus dem CIDR-Bereich des Subnetzes zugewiesen, in dem die Instance gestartet wird. In diesem Fall kann das System der primären Netzwerkschnittstelle nicht mehr automatisch eine öffentliche IPv4 Adresse zuweisen. Sie können keine Verbindung zu Ihren Instances über eine öffentliche IPv4 Adresse herstellen, es sei denn, Sie verknüpfen die verfügbaren Elastic IP-Adressen mit den Auto Scaling Scaling-Instances.

Netzwerkschnittstellen in einer VPC

Jede Instance in Ihrer VPC verfügt über eine Standard-Netzwerkschnittstelle (die primäre Netzwerkschnittstelle). Sie können eine primäre Netzwerkschnittstelle nicht von einer Instance trennen. Sie können eine zusätzliche Netzwerkschnittstelle erstellen und diese an eine Instance in Ihrer VPC anfügen. Die Anzahl der anfügbaren Netzwerkschnittstellen ist je nach Instance-Typ unterschiedlich.

Beim Starten einer Instance mithilfe einer Startvorlage können Sie zusätzliche Netzwerkschnittstellen angeben. Beim Starten einer Auto-Scaling-Instance mit mehreren Netzwerkschnittstellen wird jedoch automatisch jede Schnittstelle im selben Subnetz wie die Instance erstellt. Dies liegt daran, dass Amazon EC2 Auto Scaling die in der Startvorlage definierten Subnetze zugunsten der in der Auto-Scaling-Gruppe angegebenen Subnetze ignoriert. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).

Wenn Sie eine oder mehrere Netzwerkschnittstellen aus demselben Subnetz erstellen oder an eine Instance anfügen, kann es zu Netzwerkproblemen, z. B. asymmetrischem Routing, kommen. Dies gilt insbesondere bei Instances, die eine Variante von Linux verwenden, die nicht von Amazon stammt. Wenn Sie diese Art von Konfiguration benötigen, müssen Sie die sekundäre Netzwerkschnittstelle innerhalb des Betriebssystems konfigurieren. Ein Beispiel finden Sie unter [Wie kann ich dafür sorgen, dass meine sekundäre Netzwerkschnittstelle in meiner Ubuntu EC2-Instance funktioniert?](#) im AWS Knowledge Center.

Tenancy zur Instance-Platzierung

Standardmäßig werden alle Instances in der VPC als gemeinsame Tenancy-Instances ausgeführt. Amazon EC2 Auto Scaling unterstützt auch Dedicated Instances und Dedicated Hosts. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage mithilfe erweiterter Einstellungen](#).

AWS Outposts

AWS Outposts erweitert eine Amazon-VPC von einer AWS Region zu einem Outpost mit den VPC-Komponenten, auf die in der Region zugegriffen werden kann, darunter Internet-Gateways, virtuelle private Gateways, Amazon VPC Transit Gateways und VPC-Endpunkte. Ein Außenposten ist einer Availability Zone in der Region zugeordnet und ist eine Erweiterung dieser Availability Zone, die Sie für die Ausfallsicherheit verwenden können.

Weitere Informationen finden Sie im [AWS Outposts -Benutzerhandbuch](#).

Ein Beispiel für die Bereitstellung einer Auto-Scaling-Gruppe, die Datenverkehr von einem Application Load Balancer innerhalb eines Außenpostens bereitstellt, finden Sie im folgenden Blogbeitrag [Konfigurieren eines Application Load Balancer auf AWS Outposts](#).

Weitere Ressourcen, um mehr zu erfahren über VPCs

In den folgenden Themen erfahren Sie mehr über VPCs Subnetze.

- Private Subnetze in einer VPC
 - [Beispiel: VPC mit Servern in privaten Subnetzen und NAT](#)
 - [NAT-Gateways](#)
- Öffentliche Subnetze in einer VPC
 - [Beispiel: VPC für eine Testumgebung](#)
 - [Beispiel: VPC für Web- und Datenbankserver](#)
- Subnetze für Ihren Application Load Balancer
 - [Subnetze für Ihren Load Balancer](#)
- Allgemeine VPC-Informationen
 - [Amazon VPC User Guide](#)
 - [Stellen Sie VPCs mithilfe von VPC-Peering eine Verbindung her](#)
 - [Elastic Network-Schnittstelle](#)
 - [Verwenden Sie VPC-Endpunkte für private Konnektivität](#)

Sicherheit in Amazon EC2 Auto Scaling

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das [Modell der geteilten Verantwortung](#) beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der [AWS](#) und . Weitere Informationen zu den Compliance-Programmen, die für Amazon EC2 Auto Scaling gelten, finden Sie unter [AWS Services im Umfang nach Compliance-Programm AWS](#) .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, einschließlich der Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation zeigt Ihnen, wie Sie das Modell der übergreifenden Verantwortlichkeit bei der Verwendung von Amazon EC2 Auto Scaling einsetzen können. Die folgenden Themen veranschaulichen, wie Sie Amazon EC2 Auto Scaling zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren können. Sie erfahren auch, wie Sie andere AWS Services nutzen können, die Sie bei der Überwachung und Sicherung Ihrer Amazon EC2 Auto Scaling Scaling-Ressourcen unterstützen.

Topics

- [Infrastruktursicherheit in Amazon EC2 Auto Scaling](#)
- [Ausfallsicherheit in Amazon EC2 Auto Scaling](#)
- [Datenschutz in Amazon EC2 Auto Scaling](#)
- [Identity and Access Management für Amazon EC2 Auto Scaling](#)
- [Compliance-Validierung für Amazon EC2 Auto Scaling](#)
- [Amazon EC2 Auto Scaling und Schnittstellen-VPC-Endpunkte](#)

Infrastruktursicherheit in Amazon EC2 Auto Scaling

Als verwalteter Service ist Amazon EC2 Auto Scaling durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastructure Protection](#) in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Amazon EC2 Auto Scaling zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Sie können auch einen Virtual Private Cloud (VPC)-Endpunkt für Amazon EC2 Auto Scaling verwenden. Schnittstellen-VPC-Endpunkte ermöglichen es Ihren Amazon-VPC-Ressourcen, ihre privaten IP-Adressen zu verwenden, um auf Amazon EC2 Auto Scaling zuzugreifen, ohne dem öffentlichen Internet ausgesetzt zu sein. Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling und Schnittstellen-VPC-Endpunkte](#).

Zugehörige Ressourcen

Informationen zu Funktionen zur Isolierung des von Amazon EC2 bereitgestellten Serviceverkehrs finden Sie unter [Infrastruktursicherheit in Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch](#).

Ausfallsicherheit in Amazon EC2 Auto Scaling

Die AWS globale Infrastruktur basiert AWS-Regionen auf Availability Zones. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Gehen Sie wie folgt vor, um von der geografischen Redundanz des Availability-Zone-Designs zu profitieren:

- Verteilen Sie Ihre Auto-Scaling-Gruppe auf mehrere Availability Zones.
- Haben Sie mindestens eine funktionierende Instance in jeder Availability Zone.
- Hängen Sie einen Load Balancer an, um eingehenden Datenverkehr auf dieselben Availability Zones zu verteilen. Wenn Sie einen Application Load Balancer verwenden, stellen Sie sicher, dass jede EC2-Instance eine ähnliche Menge an Datenverkehr erhält, indem Sie zonenübergreifendes Load Balancing aktiviert lassen. Dies trägt dazu bei, die Auswirkungen einer erhöhten Last auf bestehende Instances während eines Failover-Ereignisses zu begrenzen, und führt zu einer höheren Ausfallsicherheit als ohne zonenübergreifendes Load Balancing.
- Stellen Sie sicher, dass die Elastic-Load-Balancing-Zustandsprüfungen korrekt konfiguriert sind, und dass sie für die Auto-Scaling-Gruppe aktiviert sind. Wenn eine Instance ihre Zustandsprüfung nicht besteht, wird Elastic Load Balancing den Datenverkehr nicht mehr an sie senden, sondern an fehlerfreie Instances umleiten, während Amazon EC2 Auto Scaling die fehlerhafte Instance ersetzt.

Amazon EC2 Auto Scaling unterstützt Ihre Anforderungen an die Ausfallsicherheit Ihrer Anwendungen auf folgende Weise:

- Überprüft Instances auf Zustands- und Erreichbarkeitsprobleme. Wenn eine Instance fehlerhaft wird, wird sie automatisch beendet und eine neue gestartet.
- Wenn dynamische Skalierungsrichtlinien in Kraft sind, wird die Kapazität automatisch entsprechend dem eingehenden Datenverkehr skaliert.
- Erkennt Probleme mit der Zuverlässigkeit der CloudWatch Amazon-Metriken, die Skalierungsrichtlinien unterstützen, und unterbricht Scale-In-Aktivitäten, wenn keine zuverlässigen Metriken verfügbar sind, z. B. wenn Datenpunkte fehlen.
- Versucht automatisch, die gleiche Anzahl von Instances in jeder aktivierten Availability Zone aufrechtzuerhalten, wenn Ihre Gruppe skaliert wird.
- Nutzt Availability Zones, um eine hohe Verfügbarkeit zu gewährleisten. Wenn eine Availability Zone fehlerhaft wird, geht Amazon EC2 Auto Scaling wie folgt vor:
 - Startet neue Instances in verschiedenen Availability Zones, die für Ihre Auto Scaling Scaling-Gruppe aktiviert sind.

- Verteilt Instances gleichmäßig auf alle aktivierten Availability Zones, wenn die fehlerhafte Availability Zone in einen fehlerfreien Zustand zurückkehrt.
- Versucht weiterhin, Instances in anderen aktivierten Availability Zones zu starten, wenn eine Instance in einer bestimmten Availability Zone nicht gestartet werden kann.
- Registriert und deregistriert Instances automatisch bei den Load Balancern, die Ihrer Auto-Scaling-Gruppe zugeordnet sind. So müssen Sie Instances nicht separat registrieren und deregistrieren.
- Ausfälle der Steuerungsebene für den Amazon EC2 Auto Scaling-Service APIs wirken sich nicht auf die Skalierung vorhandener Auto Scaling Scaling-Gruppen aus.

Zugehörige Ressourcen

Informationen zu den von Amazon EBS bereitgestellten Funktionen zur Unterstützung Ihrer Anforderungen an die Datenstabilität finden Sie unter [Resilience in Amazon Elastic Block Store](#) im Amazon EBS-Benutzerhandbuch.

Datenschutz in Amazon EC2 Auto Scaling

Das [Modell der AWS gemeinsamen Verantwortung](#) gilt für den Datenschutz in Amazon EC2 Auto Scaling. Wie in diesem Modell beschrieben, AWS ist es verantwortlich für den Schutz der globalen Infrastruktur, auf der AWS Cloud alle Systeme laufen. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter [Häufig gestellte Fragen zum Datenschutz](#). Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag [AWS -Modell der geteilten Verantwortung und in der DSGVO](#) im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.

- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter [Arbeiten mit CloudTrail Pfaden](#) im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-3](#).

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Amazon EC2 Auto Scaling oder anderen AWS-Services Geräten arbeiten und die Konsole AWS CLI, API oder AWS SDKs verwenden. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Wenn Sie eine Amazon EC2 EC2-Instance starten, haben Sie die Möglichkeit, Benutzerdaten an die Instance zu übergeben, um beim Booten der Instance zusätzliche Konfigurationen vorzunehmen. Wir empfehlen außerdem, niemals vertrauliche oder sensible Informationen in die Benutzerdaten aufzunehmen, die an eine Instance weitergegeben werden.

Wird AWS KMS keys zum Verschlüsseln von Amazon EBS-Volumes verwendet

Sie können Ihre Auto Scaling-Gruppe so konfigurieren, dass Amazon EBS-Volume-Daten, die in der Cloud gespeichert sind, mit AWS KMS keys verschlüsselt werden. Amazon EC2 Auto Scaling unterstützt AWS verwaltete und vom Kunden verwaltete Schlüssel zur Verschlüsselung Ihrer Daten. Beachten Sie, dass die KmsKeyId-Option zum Angeben eines kundenverwalteten Schlüssels nicht verfügbar ist, wenn Sie eine Startkonfiguration verwenden. Verwenden Sie stattdessen eine Startvorlage, um den kundenverwalteten Schlüssel anzugeben. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#). Informationen zur Erstellung,

Speicherung und Verwaltung Ihrer AWS KMS Verschlüsselungsschlüssel finden Sie im [AWS Key Management Service Entwicklerhandbuch](#).

Sie können auch einen kundenverwalteten Schlüssel in Ihrem EBS-unterstützten AMI konfigurieren, bevor Sie die Startvorlage oder die Startkonfiguration einrichten. Sie können die Verschlüsselung standardmäßig verwenden, um die Verschlüsselung der neuen EBS-Volumes und Snapshot-Kopien zu erzwingen, die Sie erstellen. Weitere Informationen finden Sie unter [Verschlüsselung mit EBS-Unterstützung verwenden AMIs](#) im Amazon EC2 EC2-Benutzerhandbuch und [Standardverschlüsselung](#) im Amazon EBS-Benutzerhandbuch.

Note

Informationen zum Einrichten der Schlüsselrichtlinie, die Sie zum Starten von Auto Scaling-Instances benötigen, wenn Sie einen kundenverwalteten Schlüssel für die Verschlüsselung verwenden, finden Sie unter [Erforderliche AWS KMS Schlüsselrichtlinie für die Verwendung mit verschlüsselten Volumes](#).

Zugehörige Ressourcen

Die von Amazon EBS bereitgestellten Datenschutzrichtlinien finden Sie unter [Datenschutz im Amazon Elastic Block Store](#) im Amazon EBS-Benutzerhandbuch.

Erforderliche AWS KMS Schlüsselrichtlinie für die Verwendung mit verschlüsselten Volumes

Amazon EC2 Auto Scaling verwendet [serviceverknüpfte Rollen](#), um Berechtigungen an andere zu delegieren. AWS-Services Servicebezogene Rollen in Amazon EC2 Auto Scaling sind vordefiniert und beinhalten Berechtigungen, die Amazon EC2 Auto Scaling benötigt, um andere AWS-Services in Ihrem Namen anzurufen. Die vordefinierten Berechtigungen beinhalten auch den Zugriff auf Ihre. Von AWS verwaltete Schlüssel Sie enthalten jedoch keinen Zugriff auf Ihre kundenverwalteten Schlüssel, sodass Sie die volle Kontrolle über diese Schlüssel behalten können.

In diesem Thema wird beschrieben, wie Sie die Schlüsselrichtlinie einrichten, die Sie zum Starten von Auto Scaling-Instances benötigen, wenn Sie einen kundenverwalteten Schlüssel für die Amazon EBS-Verschlüsselung angeben.

Note

Amazon EC2 Auto Scaling benötigt keine zusätzliche Autorisierung für die Verwendung des standardmäßigen Von AWS verwalteter Schlüssel zum Schützen der verschlüsselten Volumes in Ihrem Konto.

Inhalt

- [-Übersicht](#)
- [Konfigurieren von Schlüsselrichtlinien](#)
- [Beispiel 1: Schlüsselrichtlinienabschnitte, welche Zugriff auf den kundenverwalteten Schlüssel erlauben](#)
- [Beispiel 2: Schlüsselrichtlinienabschnitte, welche Zugriff auf den kundenverwalteten Schlüssel über mehrere Konten erlauben](#)
- [Bearbeiten Sie die wichtigsten Richtlinien in der Konsole AWS KMS](#)

-Übersicht

Folgendes AWS KMS keys kann für die Amazon EBS-Verschlüsselung verwendet werden, wenn Amazon EC2 Auto Scaling Instances startet:

- [Von AWS verwalteter Schlüssel](#)— Ein Verschlüsselungsschlüssel in Ihrem Konto, das Amazon EBS erstellt, besitzt und verwaltet. Dies ist der Standardverschlüsselungsschlüssel für ein neues Konto. Der Von AWS verwalteter Schlüssel wird für die Verschlüsselung verwendet, sofern Sie keinen vom Kunden verwalteten Schlüssel angeben.
- [Kundenverwalteter Schlüssel](#) – Ein benutzerdefinierter Verschlüsselungsschlüssel, den Sie erstellen, besitzen und verwalten. Weitere Informationen finden Sie unter [Erstellen von Schlüsseln](#) im AWS Key Management Service -Entwicklerhandbuch.

Hinweis: Der Schlüssel muss symmetrisch sein. Amazon EBS unterstützt keine asymmetrischen vom Kunden verwalteten Schlüssel.

Sie konfigurieren kundenverwaltete Schlüssel, wenn Sie verschlüsselte Snapshots oder eine Startvorlage, die verschlüsselte Volumes angibt, erstellen oder wenn Sie die Verschlüsselung standardmäßig aktivieren.

Konfigurieren von Schlüsselrichtlinien

Ihre KMS-Schlüssel müssen über eine Schlüsselrichtlinie verfügen, mit der Amazon EC2 Auto Scaling-Instances mit Amazon EBS-Volumes starten kann, die mit einem kundenverwalteten Schlüssel verschlüsselt sind.

Nutzen Sie die Beispiele auf dieser Seite zum Konfigurieren einer Schlüsselrichtlinie, um Amazon EC2 Auto Scaling Zugriff auf Ihren kundenverwalteten Schlüssel zu erteilen. Sie können die Schlüsselrichtlinie des kundenverwalteten Schlüssels entweder bei Erstellung des Schlüssels oder zu einem späteren Zeitpunkt ändern.

Sie müssen Ihrer Schlüsselrichtlinie mindestens zwei Richtlinienanweisungen hinzufügen, damit er mit Amazon EC2 Auto Scaling verwendet werden kann.

- Die erste Anweisung ermöglicht, dass die im `Principal`-Element angegebene IAM-Identität den kundenverwalteten Schlüssel direkt verwendet. Es beinhaltet Berechtigungen zur Ausführung der AWS KMS `Encrypt`, `Decrypt`, `ReEncrypt*GenerateDataKey*`, und `DescribeKey` - Operationen mit dem Schlüssel.
- Die zweite Anweisung ermöglicht es der im `Principal` Element angegebenen IAM-Identität, den `CreateGrant` Vorgang zum Generieren von Zuschüssen zu verwenden, die eine Teilmenge ihrer eigenen Berechtigungen an Personen delegieren AWS-Services , die in AWS KMS oder einen anderen `Principal` integriert sind. Auf diese Weise können sie den Schlüssel verwenden, um in Ihrem Namen verschlüsselte Ressourcen zu erstellen.

Ändern Sie beim Hinzufügen der neuen Richtlinienanweisungen zur Schlüsselrichtlinie keinen der vorhandenen Anweisungen in der Richtlinie.

Für jedes der folgenden Beispiele werden Argumente, die ersetzt werden müssen, wie z. B. eine Schlüssel-ID oder der Name einer dienstbezogenen Rolle, als angezeigt. *user placeholder text* In den meisten Fällen können Sie den Namen der serviceverknüpften Rolle durch den Namen einer serviceverknüpften Amazon EC2 Auto Scaling-Rolle ersetzen.

Weitere Informationen finden Sie in den folgenden Ressourcen:

- Informationen zum Erstellen eines Schlüssels mit dem AWS CLI finden Sie unter [create-key](#).
- Informationen zum Aktualisieren einer Schlüsselrichtlinie mit dem finden Sie AWS CLI unter. [put-key-policy](#)

- Informationen zum Ermitteln einer Schlüssel-ID und des Amazon-Ressourcennamens (ARN) finden Sie unter [Die Schlüssel-ID und den ARN finden](#) und im AWS Key Management Service - Benutzerhandbuch.
- Weitere Informationen zu Amazon EC2 Auto Scaling-Rollen finden Sie unter [Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling](#).
- [Informationen zur Amazon EBS-Verschlüsselung und KMS im Allgemeinen sowie zur Amazon EBS-Verschlüsselung finden Sie im Amazon EBS-Benutzerhandbuch und im AWS Key Management Service Entwicklerhandbuch.](#)

Beispiel 1: Schlüsselrichtlinienabschnitte, welche Zugriff auf den kundenverwalteten Schlüssel erlauben

Fügen Sie der Schlüsselrichtlinie des kundenverwalteten Schlüssel die folgenden beiden Richtlinienanweisungen hinzu und ersetzen Sie dabei den Beispiel ARN durch den ARN der entsprechenden serviceverknüpften Rolle, der Zugriff auf den Schlüssel gewährt wird. In diesem Beispiel gewähren Richtlinienabschnitte der serviceverknüpften Rolle mit dem Namen `AWSServiceRoleForAutoScaling` Berechtigungen zur Verwendung des kundenverwalteten Schlüssel.

```
{
  "Sid": "Allow service-linked role use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

```
{
  "Sid": "Allow attachment of persistent resources",
```

```
"Effect": "Allow",
"Principal": {
  "AWS": [
    "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
  ]
},
"Action": [
  "kms:CreateGrant"
],
"Resource": "*",
"Condition": {
  "Bool": {
    "kms:GrantIsForAWSResource": true
  }
}
}
```

Beispiel 2: Schlüsselrichtlinienabschnitte, welche Zugriff auf den kundenverwalteten Schlüssel über mehrere Konten erlauben

Wenn Sie einen vom Kunden verwalteten Schlüssel in einem anderen Konto als der Auto-Scaling-Gruppe erstellen, müssen Sie eine Berechtigung in Kombination mit der Schlüsselrichtlinie verwenden, um den kontoübergreifenden Zugriff auf den Schlüssel zu ermöglichen.

Es gibt zwei Schritte, die in der folgenden Reihenfolge ausgeführt werden müssen:

1. Fügen Sie zunächst die folgenden beiden Richtlinienerklärungen zur Schlüsselrichtlinie des vom Kunden verwalteten Schlüssels hinzu. Ersetzen Sie den Beispiel-ARN durch den ARN des anderen Kontos und achten Sie darauf, ihn durch die tatsächliche Konto-ID des Kontos zu **111122223333** ersetzen, in dem Sie AWS-Konto die Auto Scaling Scaling-Gruppe erstellen möchten. Damit können Sie einem IAM-Benutzer oder einer IAM-Rolle im angegebenen Konto die Berechtigung erteilen, mit dem folgenden CLI-Befehl eine Berechtigung für den Schlüssel zu erstellen. Dies gewährt jedoch keinen Benutzern Zugriff auf den Schlüssel.

```
{
  "Sid": "Allow external account 111122223333 use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:root"
    ]
  }
}
```

```

    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}

```

```

{
  "Sid": "Allow attachment of persistent resources in external
account 111122223333",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:root"
    ]
  },
  "Action": [
    "kms:CreateGrant"
  ],
  "Resource": "*"
}

```

- Erstellen Sie dann von dem Konto aus, in dem Sie die Auto-Scaling-Gruppe erstellen möchten, eine Berechtigung, welche die relevanten Berechtigungen an die entsprechende serviceverknüpfte Rolle delegiert. Das Grantee Principal-Element der Berechtigung ist der ARN der dazugehörigen serviceverknüpften Rolle. key-id ist der ARN des Schlüssels.

Im Folgenden finden Sie ein Beispiel für einen [Create-Grant-CLI-Befehl](#), der der im Konto genannten dienstverknüpften Rolle die **111122223333** Berechtigung erteilt, den vom Kunden verwalteten Schlüssel **AWSServiceRoleForAutoScaling** im Konto zu verwenden. **444455556666**

```

aws kms create-grant \
  --region us-west-2 \
  --key-id arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \
  --grantee-principal arn:aws:iam::<111122223333>:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling \

```

```
--operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"
"GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
```

Damit dieser Befehl erfolgreich ist, muss der Benutzer, der die Anforderung stellt, über Berechtigungen für die `CreateGrant`-Aktion verfügen.

Das folgende Beispiel für eine IAM-Richtlinie ermöglicht es einer IAM-Identität (Benutzer oder Rolle) in einem Konto, einen Zuschuss für das vom Kunden verwaltete Key-in-Konto **111122223333** zu erstellen. **444455556666**

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreationOfGrantForTheKMSKeyinExternalAccount444455556666",
      "Effect": "Allow",
      "Action": "kms:CreateGrant",
      "Resource": "arn:aws:kms:us-west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    }
  ]
}
```

Weitere Informationen über die Erstellung eines Zuschusses für einen KMS-Schlüssel in einem anderen AWS-Konto, finden Sie unter [Berechtigungsverteilungen in AWS KMS](#) im AWS Key Management Service -Entwicklerhandbuch.

Important

Der Name der serviceverknüpften Rolle, der als Prinzipal des Empfängers angegeben wird, muss der Name einer vorhandenen Rolle sein. Um nach dem Erstellen der Erteilung sicherzustellen, dass die Erteilung Amazon EC2 Auto Scaling erlaubt, den angegebenen KMS-Schlüssel zu verwenden, löschen Sie die serviceverknüpfte Rolle nicht und erstellen Sie diese nicht neu.

Bearbeiten Sie die wichtigsten Richtlinien in der Konsole AWS KMS

Die Beispiele in den vorherigen Abschnitten zeigen nur, wie einer Schlüsselrichtlinie Anweisungen hinzugefügt werden, was nur eine Möglichkeit darstellt, eine Schlüsselrichtlinie zu ändern. Die einfachste Möglichkeit, eine Schlüsselrichtlinie zu ändern, besteht darin, die Standardansicht der AWS KMS Konsole für wichtige Richtlinien zu verwenden und eine IAM-Identität (Benutzer oder Rolle) zu einem der Hauptbenutzer für die entsprechende Schlüsselrichtlinie zu machen. Weitere Informationen finden Sie im AWS Key Management Service Entwicklerhandbuch [unter Verwenden der AWS-Managementkonsole Standardansicht](#).

Important

Gehen Sie vorsichtig vor. Die Standardansichtsrichtlinien der Konsole beinhalten Berechtigungen zur Ausführung von AWS KMS Revoke Vorgängen mit dem vom Kunden verwalteten Schlüssel. Wenn Sie AWS-Konto Zugriff auf einen vom Kunden verwalteten Schlüssel in Ihrem Konto gewähren und Sie versehentlich die Erteilung widerrufen, mit der sie ihnen diese Berechtigung erteilt haben, können externe Benutzer nicht mehr auf ihre verschlüsselten Daten oder den Schlüssel, der zur Verschlüsselung ihrer Daten verwendet wurde, zugreifen.

Identity and Access Management für Amazon EC2 Auto Scaling

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf AWS Ressourcen sicher zu kontrollieren. IAM-Administratoren steuern, wer authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) ist, Amazon EC2 Auto Scaling-Ressourcen zu nutzen. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Um Amazon EC2 Auto Scaling verwenden zu können, benötigen Sie eine AWS-Konto und Ihre Sicherheitsanmeldedaten für die Anmeldung bei Ihrem Konto. Weitere Informationen finden Sie unter [AWS Sicherheitsanmeldedaten](#) im IAM-Benutzerhandbuch.

Eine umfassende IAM-Dokumentation finden Sie im [IAM User Guide](#).

Zugriffskontrolle

Auch wenn Sie über gültige Anmeldeinformationen zur Authentifizierung Ihrer Anfragen verfügen, können Sie die Amazon EC2 Auto Scaling-Ressourcen nur mit entsprechenden Berechtigungen erstellen oder darauf zugreifen. Beispielsweise müssen Sie über Berechtigungen zum Erstellen von Auto-Scaling-Gruppen, zum Starten von Instances mit Startvorlagen usw. verfügen.

Dieses Thema enthält Informationen dazu, wie ein IAM-Administrator Ihre Amazon EC2 Auto Scaling-Ressourcen mithilfe von IAM sichern kann, indem er steuert, wer Amazon EC2 Auto Scaling-Aktionen durchführen darf.

Wir empfehlen Ihnen, die Amazon EC2-Themen zuerst zu lesen. Weitere Informationen finden Sie unter [Identitäts- und Zugriffsmanagement für Amazon EC2](#) im Amazon EC2-Benutzerhandbuch. Nachdem Sie die Themen in diesem Abschnitt gelesen haben, sollten Sie eine gute Vorstellung davon haben, welche Zugriffsberechtigungen Amazon EC2 bietet, und wie diese zu Ihren Amazon EC2 Auto Scaling-Ressourcenberechtigungen passen können.

Themen

- [Funktionsweise von Amazon EC2 Auto Scaling mit IAM](#)
- [API-Berechtigungen für Amazon EC2 Auto Scaling](#)
- [AWS verwaltete Richtlinien für Amazon EC2 Auto Scaling](#)
- [Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling](#)
- [Beispiele für identitätsbasierte Amazon EC2 Auto Scaling-Richtlinien](#)
- [Serviceübergreifende Confused-Deputy-Prävention](#)
- [Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling-Gruppen](#)
- [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#)

Funktionsweise von Amazon EC2 Auto Scaling mit IAM

Informieren Sie sich zunächst darüber, welche IAM-Features mit Amazon EC2 Auto Scaling verwendet werden können, bevor Sie IAM verwenden, um den Zugriff auf Amazon EC2 Auto Scaling zu verwalten.

IAM-Features, die mit Amazon EC2 Auto Scaling verwendet werden können

IAM-Feature	Amazon EC2 Auto Scaling-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (servicespezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Teilweise
Temporäre Anmeldeinformationen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Ja

Einen allgemeinen Überblick darüber, wie Amazon EC2 Auto Scaling und andere AWS-Services mit den meisten IAM-Funktionen [funktionieren AWS-Services](#), finden Sie im [IAM-Benutzerhandbuch](#).

Identitätsbasierte Richtlinien für Amazon EC2 Auto Scaling

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien](#) im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter

denen Aktionen zugelassen oder abgelehnt werden. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der [IAM-Referenz für JSON-Richtlinienelemente](#) im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien in Amazon EC2 Auto Scaling

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#). Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Weitere Informationen finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Richtlinienaktionen für Amazon EC2 Auto Scaling

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu erteilen.

Eine Liste der Amazon EC2 Auto Scaling-Aktionen finden Sie in der Service Authorization Reference unter [Von Amazon EC2 Auto Scaling definierte Aktionen](#).

Bei Richtlinienaktionen in Amazon EC2 Auto Scaling wird der Aktion das folgende Präfix vorangestellt:

```
autoscaling
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [  
  "autoscaling:action1",  
  "autoscaling:action2"  
]
```

Mithilfe von Platzhaltern (*) können mehrere Aktionen angegeben werden. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "autoscaling:Describe*
```

Richtlinienressourcen für Amazon EC2 Auto Scaling

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen [Amazon-Ressourcennamen \(ARN\)](#) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*" 
```

Sie können ARNs damit die Auto Scaling Scaling-Gruppen und Startkonfigurationen identifizieren, für die die IAM-Richtlinie gilt.

Eine Auto Scaling-Gruppe hat den folgenden ARN.

```
"Resource": "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
```

Eine Startkonfiguration hat den folgenden ARN.

```
"Resource": "arn:aws:autoscaling:region:account-id:launchConfiguration:uuid:launchConfigurationName/lc-name"
```

Um eine Auto-Scaling-Gruppe mit der Aktion `CreateAutoScalingGroup` anzugeben, müssen Sie die UUID durch einen Platzhalter (*) ersetzen, wie im folgenden Beispiel zu sehen.

```
"Resource": "arn:aws:autoscaling:region:account-  
id:autoScalingGroup:*:autoScalingGroupName/asg-name"
```

Um eine Startkonfiguration mit der Aktion `CreateLaunchConfiguration` anzugeben, müssen Sie die UUID durch einen Platzhalter (*) ersetzen, wie im folgenden Beispiel zu sehen.

```
"Resource": "arn:aws:autoscaling:region:account-  
id:launchConfiguration:*:launchConfigurationName/lc-name"
```

Weitere Informationen zu Amazon EC2 Auto Scaling-Ressourcentypen und deren ARNs Typen finden Sie unter [Von Amazon EC2 Auto Scaling definierte Ressourcen](#) in der Service Authorization Reference. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter [Von Amazon EC2 Auto Scaling definierte Aktionen](#).

Note

Ein Beispiel für eine IAM-Richtlinie, mit der der Zugriff auf Auto Scaling Scaling-Gruppen gesteuert wird, finden Sie unter [Steuern Sie, welche Auto-Scaling-Gruppen gelöscht werden können](#). ARNs

Nicht alle Amazon EC2 Auto Scaling-Aktionen unterstützen Berechtigungen auf Ressourcenebene. Für Aktionen, die Berechtigungen auf Ressourcenebene nicht unterstützen, muss ein Platzhalter (*) als Ressource verwendet werden.

Die folgenden Amazon EC2 Auto Scaling-Aktionen unterstützen keine Berechtigungen auf Ressourcenebene.

- `DescribeAccountLimits`
- `DescribeAdjustmentTypes`
- `DescribeAutoScalingGroups`
- `DescribeAutoScalingInstances`
- `DescribeAutoScalingNotificationTypes`

- DescribeInstanceRefreshes
- DescribeLaunchConfigurations
- DescribeLifecycleHooks
- DescribeLifecycleHookTypes
- DescribeLoadBalancers
- DescribeLoadBalancerTargetGroups
- DescribeMetricCollectionTypes
- DescribeNotificationConfigurations
- DescribePolicies
- DescribeScalingActivities
- DescribeScalingProcessTypes
- DescribeScheduledActions
- DescribeTags
- DescribeTerminationPolicyTypes
- DescribeWarmPool

Richtlinienbedingungsschlüssel für Amazon EC2 Auto Scaling

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die [Bedingungsoperatoren](#) verwenden, z. B. `ist gleich` oder `kleiner als`, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter [Kontextschlüssel für AWS globale Bedingungen](#) im IAM-Benutzerhandbuch.

Amazon EC2 Auto Scaling unterstützt die folgenden Bedingungsschlüssel, die verwendet werden können, um den Zugriff auf unterstützte Aktionen zu kontrollieren und die Konfiguration von Auto-Scaling-Gruppen zu erzwingen:

- `autoscaling:InstanceTypes`

- `autoscaling:LaunchConfigurationName`
- `autoscaling:LaunchTemplateVersionSpecified`
- `autoscaling:LoadBalancerNames`
- `autoscaling:MaxSize`
- `autoscaling:MinSize`
- `autoscaling:ResourceTag/key-name: tag-value`
- `autoscaling:TargetGroupARNs`
- `autoscaling:VPCZoneIdentifiers`
- `autoscaling:ForceDelete`

Die folgenden Bedingungsschlüssel gelten speziell für die Erstellung von Startkonfigurationsanforderungen:

- `autoscaling:ImageId`
- `autoscaling:InstanceType`
- `autoscaling:MetadataHttpEndpoint`
- `autoscaling:MetadataHttpPutResponseHopLimit`
- `autoscaling:MetadataHttpTokens`
- `autoscaling:SpotPrice`

Amazon EC2 Auto Scaling unterstützt auch die folgenden globalen Bedingungsschlüssel, mit denen Sie Berechtigungen basierend auf den Tags in der Anforderung definieren können oder die in der Auto Scaling-Gruppe vorhanden sind. Weitere Informationen finden Sie unter [Tagging von Auto-Scaling-Gruppen und Instances](#).

- `aws:RequestTag/key-name: tag-value`
- `aws:ResourceTag/key-name: tag-value`
- `aws:TagKeys: [tag-key, ...]`

Eine Liste der Amazon EC2 Auto Scaling-Aktionen finden Sie in der Service Authorization Reference unter [Von Amazon EC2 Auto Scaling definierte Aktionen](#). Weitere Informationen zu Amazon EC2 Auto Scaling-Bedingungsschlüsseln finden Sie unter [Bedingungsschlüssel für Amazon EC2 Auto Scaling](#).

Note

Beispiele für IAM-Richtlinien, die Bedingungsschlüssel verwenden, um den Zugriff auf unterstützte Aktionen zu kontrollieren und die Konfiguration von Auto-Scaling-Gruppen zu erzwingen, finden Sie in den folgenden Ressourcen:

- [Eine Startvorlage und eine Versionsnummer verlangen](#)— In diesem Beispiel wird erzwungen, dass beim Erstellen oder Aktualisieren von Auto Scaling Scaling-Gruppen eine Startvorlage und die Versionsnummer der Startvorlage angegeben werden müssen.
- [Steuern Sie die Größe der Auto-Scaling-Gruppen, die erstellt werden können](#)— In diesem Beispiel werden Beschränkungen für die möglichen Werte für die MaxSize Eigenschaften MinSize und beim Erstellen oder Aktualisieren von Auto Scaling Scaling-Gruppen mit einem bestimmten Tag erzwungen.
- [Steuern, welche Skalierungsrichtlinien gelöscht werden können](#)— In diesem Beispiel wird erzwungen, dass das Löschen von Skalierungsrichtlinien nur für Auto Scaling Scaling-Gruppen ohne ein bestimmtes Tag zulässig ist.

ACLs in Amazon EC2 Auto Scaling

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Amazon EC2 Auto Scaling

Unterstützt ABAC (Tags in Richtlinien): Teilweise

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen, auch als Tags bezeichnet, definiert werden. Sie können Tags an IAM-Entitäten und AWS Ressourcen anhängen und dann ABAC-Richtlinien so entwerfen, dass Operationen möglich sind, wenn das Tag des Prinzipals mit dem Tag auf der Ressource übereinstimmt.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungelement einer [Richtlinie Tag-Informationen](#) an, indem Sie die Schlüssel `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter [Definieren von Berechtigungen mit ABAC-Autorisierung](#) im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe [Attributbasierte Zugriffskontrolle \(ABAC\)](#) verwenden im IAM-Benutzerhandbuch.

ABAC ist für Ressourcen möglich, die Tags unterstützen. Tags werden jedoch nicht von allen Ressourcen unterstützt. Startkonfigurationen und Skalierungsrichtlinien unterstützen keine Tags, Auto-Scaling-Gruppen dagegen schon.

Weitere Informationen finden Sie unter [Tagging von Auto-Scaling-Gruppen und Instances](#).

Verwenden temporärer Anmeldeinformationen mit Amazon EC2 Auto Scaling

Unterstützt temporäre Anmeldeinformationen: Ja

Temporäre Anmeldeinformationen ermöglichen den kurzfristigen Zugriff auf AWS Ressourcen und werden automatisch erstellt, wenn Sie einen Verbund verwenden oder die Rollen wechseln. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter [Temporäre Anmeldeinformationen in IAM](#) und [AWS-Services , die mit IAM funktionieren](#) im IAM-Benutzerhandbuch.

Servicerollen für Amazon EC2 Auto Scaling

Unterstützt Servicerollen: Ja

Eine Servicerolle ist eine [IAM-Rolle](#), die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter [Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service](#) im IAM-Benutzerhandbuch.

Wenn Sie einen Lebenszyklus-Hook erstellen, der ein Amazon SNS-Thema oder eine Amazon SQS-Warteschlange benachrichtigt, müssen Sie eine Rolle angeben, damit Amazon EC2 Auto Scaling in Ihrem Namen auf Amazon SNS oder Amazon SQS zugreifen kann. Verwenden Sie die IAM-Konsole, um die Servicerolle für Ihren Lebenszyklus-Hook einzurichten. Die Konsole unterstützt Sie bei der Erstellung einer Rolle mit ausreichenden Berechtigungen mit einer verwalteten Richtlinie.

Weitere Informationen erhalten Sie unter [Benachrichtigungen über Amazon SNS erhalten](#) und [Benachrichtigungen über Amazon SQS erhalten](#).

Wenn Sie eine Auto Scaling Scaling-Gruppe erstellen, können Sie optional eine Servicerolle übergeben, damit Amazon EC2 EC2-Instances in Ihrem Namen AWS-Services auf andere zugreifen können. Die Servicerolle für Amazon EC2-Instances (auch Amazon EC2-Instance-Profil für eine Startvorlage oder Startkonfiguration genannt) ist eine spezielle Art von Servicerolle, die jeder EC2-Instance in einer Auto-Scaling-Gruppe zugewiesen wird, wenn die Instance startet. Sie können die IAM-Konsole verwenden und AWS CLI diese Servicerolle erstellen oder bearbeiten. Weitere Informationen finden Sie unter [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#).

Warning

Das Ändern der Berechtigungen für eine Servicerolle kann dazu führen, dass Amazon EC2 Auto Scaling nicht mehr funktioniert. Bearbeiten Sie Servicerollen nur, wenn Sie von Amazon EC2 Auto Scaling dazu angeleitet werden.

Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling

Unterstützt serviceverknüpfte Rollen: Ja

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einer Service-Verknüpfung ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Amazon EC2 Auto Scaling-Rollen finden Sie unter [Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling](#).

API-Berechtigungen für Amazon EC2 Auto Scaling

Sie müssen den Benutzern die Berechtigung erteilen, die benötigten Amazon EC2 Auto Scaling-API-Aktionen aufzurufen, wie in [Richtlinienaktionen für Amazon EC2 Auto Scaling](#) beschrieben. Darüber hinaus müssen Sie Benutzern für einige Amazon EC2 Auto Scaling Scaling-Aktionen die Erlaubnis erteilen, bestimmte Aktionen von anderen AWS APIs aus aufzurufen.

Erforderliche Berechtigungen von anderen AWS APIs

Zusätzlich zu den Amazon EC2 Auto Scaling Scaling-API-Berechtigungen müssen Benutzer über die folgenden Berechtigungen von anderen verfügen, AWS APIs um die zugehörige Aktion erfolgreich ausführen zu können.

Erstellen einer Auto-Scaling-Gruppe (`autoscaling:CreateAutoScalingGroup`)

- `iam:CreateServiceLinkedRole`— Um die standardmäßige serviceverknüpfte Rolle zu erstellen, falls diese Rolle noch nicht existiert.
- `iam:PassRole`— Um beim Start eine IAM-Rolle an den Service oder an EC2-Instances zu übergeben. Wird benötigt, wenn eine nicht standardmäßige serviceverknüpfte Rolle, eine IAM-Rolle für einen Lebenszyklus-Hook oder eine Startvorlage, die ein Instance-Profil spezifiziert (ein Container für eine IAM-Rolle), bereitgestellt wird.
- `ec2:RunInstances`— Um Instances zu starten, wenn eine Startvorlage bereitgestellt wird.
- `ec2:CreateTags`— Um Instances und Volumes beim Start zu taggen, wenn eine Startvorlage mit einer Tag-Spezifikation bereitgestellt wird.

Erstellen eines Lebenszyklus-Hooks (`autoscaling:PutLifecycleHook`)

- `iam:PassRole`— Um eine IAM-Rolle an den Service zu übergeben. Wird benötigt, wenn eine IAM-Rolle bereitgestellt wird.

Fügen Sie eine VPC-Lattice-Zielgruppe hinzu (`autoscaling:AttachTrafficSources`)

- `vpc-lattice:RegisterTargets`— Um Instanzen automatisch bei der Zielgruppe zu registrieren.

Eine VPC-Lattice-Zielgruppe abtrennen (`autoscaling:DetachTrafficSources`)

- `vpc-lattice:DeregisterTargets`— Um Instanzen automatisch bei der Zielgruppe zu deregistrieren.

Erstellen einer Startkonfiguration (`autoscaling>CreateLaunchConfiguration`)

- `ec2:DescribeImages`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceAttribute`
- `ec2:DescribeKeyPairs`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSpotInstanceRequests`
- `ec2:DescribeVpcClassicLink`

- `iam:PassRole`— Um beim Start eine IAM-Rolle an EC2-Instances zu übergeben. Erforderlich, wenn eine Startkonfiguration ein Instance-Profil (einen Container für eine IAM-Rolle) angibt.

AWS verwaltete Richtlinien für Amazon EC2 Auto Scaling

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet AWS wird. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige Anwendungsfälle bereitstellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie [vom Kunden verwaltete Richtlinien](#) definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter [Von AWS verwaltete Richtlinien](#) im IAM-Benutzerhandbuch.

Von Amazon EC2 Auto Scaling verwaltete Richtlinien

Sie können die folgenden verwalteten Richtlinien an Ihre AWS Identity and Access Management (IAM-) Identitäten (Benutzer oder Rollen) anhängen. Jede Richtlinie gewährt Zugriff auf alle oder einige der API-Aktionen für Amazon EC2 Auto Scaling.

- [AutoScalingConsoleFullAccess](#)— Gewährt vollen Zugriff auf Amazon EC2 Auto Scaling mithilfe von. AWS-Managementkonsole Diese Richtlinie funktioniert, wenn Sie Startkonfigurationen verwenden, aber nicht, wenn Sie Startvorlagen verwenden.
- [AutoScalingConsoleReadOnlyAccess](#)— Gewährt schreibgeschützten Zugriff auf Amazon EC2 Auto Scaling mithilfe von. AWS-Managementkonsole Diese Richtlinie funktioniert, wenn Sie Startkonfigurationen verwenden, aber nicht, wenn Sie Startvorlagen verwenden.
- [AutoScalingFullAccess](#)— Gewährt vollen Zugriff auf Amazon EC2 Auto Scaling für IAM-Identitäten, die vollen Amazon EC2 Auto Scaling Scaling-Zugriff über AWS CLI oder benötigen SDKs, aber keinen Zugriff. AWS-Managementkonsole

- [AutoScalingReadOnlyAccess](#)— Gewährt Nur-Lese-Zugriff auf Amazon EC2 Auto Scaling für IAM-Identitäten, die nur das Oder aufrufen. AWS CLI SDKs

Bei der Verwendung von Startvorlagen über die Konsole müssen Sie zusätzliche, für Startvorlagen spezifische Berechtigungen erteilen, die unter [Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling-Gruppen](#) erläutert werden. Die Amazon EC2 Auto Scaling-Konsole benötigt Berechtigungen für ec2-Aktionen, damit Informationen über Startvorlagen angezeigt und Instances mithilfe von Startvorlagen gestartet werden können.

AutoScalingServiceRolePolicy AWS verwaltete Richtlinie

Diese Richtlinie ist mit einer servicebezogenen Rolle verknüpft, die es Amazon EC2 Auto Scaling ermöglicht, Aktionen in Ihrem Namen durchzuführen. Weitere Informationen finden Sie unter [Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling](#).

Informationen zu den Berechtigungen für diese Richtlinie finden Sie unter [AutoScalingServiceRolePolicy](#) in der Referenz zu von AWS verwalteten Richtlinien.

Amazon EC2 Auto Scaling Scaling-Updates für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Amazon EC2 Auto Scaling an, seit dieser Service begonnen hat, diese Änderungen zu verfolgen. Um automatische Warnungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der Amazon-EC2-Auto-Scaling Seite „Document history“ (Dokumentverlauf).

Änderungen	Beschreibung	Date
Amazon EC2 Auto Scaling aktualisiert die Berechtigungen für seine serviceverknüpfte Rolle	Die AutoScalingService RolePolicy Richtlinie umfasst jetzt die ec2:CreateReplaceRootVolume Task Berechtigung, die es Ihnen ermöglicht, Root-Volumes auf Instances im Rahmen einer Instance-Aktualisierung zu ersetzen. Darüber hinaus wurde die events:DescribeRule Berechtig	18. November 2025

Änderungen	Beschreibung	Date
	<p>ung aus der Richtlinie entfernt. Weitere Informationen finden Sie unter Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling.</p>	
Amazon EC2 Auto Scaling fügt seiner service-verknüpften Rolle Berechtigungen zu	<p>Die <code>AutoScalingServiceRolePolicy</code> Richtlinie beinhaltet jetzt die Erlaubnis, die AWS -Ressourcengruppen ListGroupResources API-Aktion aufzurufen, um alle Ressourcennamen (ARNs) der Ressourcen abzurufen, die Mitglieder einer bestimmten Ressourcengruppe sind. Weitere Informationen finden Sie unter Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling.</p>	20. November 2024

Änderungen	Beschreibung	Date
Amazon EC2 Auto Scaling fügt seiner service-verknüpften Rolle Berechtigungen zu	Die AutoScalingService RolePolicy Richtlinie gewährt nun Berechtigungen zum Aufrufen der Amazon EC2 GetSecurityGroupsForVpc EC2-API-Aktion, um alle Sicherheitsgruppen für eine VPC abzurufen, um die Validierung zu verbessern, und der Amazon EC2 GetInstanceTypesFromInstanceRequirements EC2-API-Aktion, um Informationen darüber abzurufen, welche Instance-Typen bestimmte Instance-Anforderungen erfüllen. Weitere Informationen finden Sie unter Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling .	29. Februar 2024

Änderungen	Beschreibung	Date
Amazon EC2 Auto Scaling fügt seiner service-verknüpften Rolle Berechtigungen zu	<p>Die Richtlinie <code>AutoScalingServiceRolePolicy</code> gewährt dem Service nun Berechtigungen für den Zugriff auf die API-Aktionen, die er für eine Integration mit VPC Lattice benötigt.</p> <ul style="list-style-type: none">• <code>GetTargetGroup</code> - und <code>ListTargetGroup</code> - Aktionen. Erforderlich zum Abrufen von Informationen zu den VPC-Lattice-Zielgruppen.• <code>RegisterTargets</code> - und <code>DeregisterTargets</code> - Aktionen. Erforderlich zum Registrieren und Deregistrieren von Instances bei VPC-Lattice-Zielgruppen.• <code>ListTargets</code> . Ermöglicht Amazon EC2 Auto Scaling das Abrufen von Zustandsinformationen für Instances , die bei VPC-Lattice-Zielgruppen registriert sind. <p>Weitere Informationen finden Sie unter Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling.</p>	6. Dezember 2022

Änderungen	Beschreibung	Date
Amazon EC2 Auto Scaling fügt seiner service-verknüpften Rolle Berechtigungen zu	Um die Verwendung eines AWS Systems Manager Parameters als Alias für eine AMI-ID beim Erstellen einer Startvorlage zu unterstützen, gewährt die <code>AutoScalingServiceRolePolicy</code> Richtlinie jetzt die Erlaubnis, die AWS Systems Manager GetParametersAPI -Aktion aufzurufen. Weitere Informationen finden Sie unter Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling .	28. März 2022
Amazon EC2 Auto Scaling fügt seiner service-verknüpften Rolle Berechtigungen zu	Zur Unterstützung der prädiktiven Skalierung enthält die <code>AutoScalingServiceRolePolicy</code> -Richtlinie jetzt die Berechtigung zum Aufrufen der CloudWatch GetMetricData API-Aktion. Weitere Informationen finden Sie unter Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling .	19. Mai 2021
Amazon EC2 Auto Scaling startet Nachverfolgung von Änderungen	Amazon EC2 Auto Scaling begann, Änderungen an seinen AWS verwalteten Richtlinien nachzuverfolgen.	19. Mai 2021

Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling nutzt serviceverknüpfte Rollen für die Berechtigungen, die für den Aufruf anderer AWS-Services -Services in Ihrem Namen benötigt werden. Eine serviceverknüpfte Rolle ist eine einzigartige Art von IAM-Rolle, die direkt mit einer verknüpft ist. AWS-Service

Serviceverknüpfte Rollen bieten eine sichere Möglichkeit, um Berechtigungen zu AWS-Services -Services zu delegieren, da nur der verknüpfte Service eine serviceverknüpfte Rolle annehmen kann. Weitere Informationen finden Sie im [IAM-Benutzerhandbuch unter Erstellen einer serviceverknüpften Rolle](#). Serviceverknüpfte Rollen ermöglichen außerdem, dass alle API-Aufrufe sichtbar sind. AWS CloudTrail Das hilft bei Überwachungs- und Prüfungsanforderungen, da Sie alle in Ihrem Namen von Amazon EC2 Auto Scaling ausgeführten Aktionen nachverfolgen können. Weitere Informationen finden Sie unter [Protokollieren Sie Amazon EC2 Auto Scaling API-Aufrufe mit AWS CloudTrail](#).

In den folgenden Abschnitten wird die Erstellung und Verwaltung von serviceverknüpften Amazon EC2 Auto Scaling-Rollen beschrieben. Beginnen Sie mit dem Konfigurieren von Berechtigungen, damit eine IAM-Identität (z. B. ein Benutzer oder eine Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann.

Inhalt

- [-Übersicht](#)
- [Von der serviceverknüpften Rolle erteilte Berechtigungen](#)
- [Unterstützte Regionen für Amazon EC2 Auto Scaling serviceverknüpfte Rollen](#)
- [Eine serviceverknüpfte Rolle erstellen, bearbeiten und löschen](#)
 - [Erstellen einer serviceverknüpften Rolle \(automatisch\)](#)
 - [Erstellen einer serviceverknüpften Rolle \(manuell\)](#)
 - [Bearbeiten der serviceverknüpften Rolle](#)
 - [Löschen der serviceverknüpften Rolle](#)

-Übersicht

Es gibt zwei Typen von serviceverknüpften Amazon EC2 Auto Scaling-Rollen:

- Die standardmäßige serviceverknüpfte Rolle für Ihr Konto, mit dem Namen `AWSServiceRoleForAutoScaling`. Diese Rolle wird automatisch Ihren Auto Scaling-Gruppen zugewiesen, es sei denn, Sie geben eine andere serviceverknüpfte Rolle an.

- Eine serviceverknüpfte Rolle mit einem benutzerdefinierten Suffix, das Sie beim Erstellen der Rolle angeben, z. B. `AWSServiceRoleForAutoScaling_mysuffix`.

Eine serviceverknüpfte Rolle mit benutzerdefiniertem Suffix hat dieselben Berechtigungen wie die standardmäßige serviceverknüpfte Rolle. In beiden Fällen können Sie die Rollen nicht bearbeiten und auch nicht löschen, wenn sie noch von einer Auto Scaling-Gruppe verwendet werden. Der einzige Unterschied ist das Suffix des Rollennamens.

Sie können beide Rollen angeben, wenn Sie Ihre AWS Key Management Service Schlüsselrichtlinien bearbeiten, sodass Instances, die von Amazon EC2 Auto Scaling gestartet werden, mit Ihrem vom Kunden verwalteten Schlüssel verschlüsselt werden können. Wenn Sie jedoch vorhaben, einem bestimmten kundenverwalteten Schlüssel individuell Zugriff zu gewähren, sollten Sie eine serviceverknüpfte Rolle mit benutzerdefiniertem Suffix verwenden. Eine serviceverknüpfte Rolle mit benutzerdefiniertem Suffix bietet Ihnen:

- Mehr Kontrolle über den kundenverwalteten Schlüssel
- Die Möglichkeit, in Ihren CloudTrail Protokollen nachzuverfolgen, welche Auto Scaling Scaling-Gruppe einen API-Aufruf getätigt hat

Wenn Sie kundenverwaltete Schlüssel erstellen, auf die nicht alle Benutzer Zugriff haben sollen, führen Sie diese Schritte aus, um die Verwendung einer serviceverknüpften Rolle mit benutzerdefiniertem Suffix zuzulassen:

1. Erstellen Sie eine serviceverknüpfte Rolle mit einem benutzerdefinierten Suffix. Weitere Informationen finden Sie unter [Erstellen einer serviceverknüpften Rolle \(manuell\)](#).
2. Erteilen Sie der serviceverknüpften Rolle Zugriff auf einen kundenverwalteten Schlüssel. Weitere Informationen über die Schlüsselrichtlinie, die zulässt, dass der Schlüssel von einer serviceverknüpften Rolle verwendet wird, finden Sie unter [Erforderliche AWS KMS Schlüsselrichtlinie für die Verwendung mit verschlüsselten Volumes](#).
3. Geben Sie Benutzern Zugriff auf die von Ihnen erstellte serviceverknüpfte Rolle. Weitere Informationen zum Erstellen der IAM-Richtlinie finden Sie unter [Steuern Sie, welche serviceverknüpfte Rolle übergeben werden kann \(mit\) PassRole](#). Wenn Benutzer versuchen, eine serviceverknüpfte Rolle ohne Berechtigung anzugeben, diese Rolle an den Service weiterzugeben, wird eine Fehlermeldung angezeigt.

Von der serviceverknüpften Rolle erteilte Berechtigungen

Amazon EC2 Auto Scaling verwendet die benannte serviceverknüpfte Rolle `AWSServiceRoleForAutoScaling` oder Ihr benutzerdefiniertes Suffix für serviceverknüpfte Rolle.

Die serviceverknüpfte Rolle vertraut darauf, dass der folgende Service die Rolle annimmt:

- `autoscaling.amazonaws.com`

Die Rollenberechtigungsrichtlinie, [AutoScalingServiceRolePolicy](#), ermöglicht Amazon EC2 Auto Scaling, die folgenden Aktionen durchzuführen:

- `ec2`— EC2-Instances erstellen, beschreiben, ändern, starten/stoppen und beenden.
- `iam`— [Übergeben Sie IAM-Rollen](#) an EC2-Instances, sodass Anwendungen, die auf den Instances ausgeführt werden, auf temporäre Anmeldeinformationen für die Rolle zugreifen können.
- `iam`— Erstellen Sie die mit dem `AWSServiceRoleForEC2Spot`-Dienst verknüpfte Rolle, damit Amazon EC2 Auto Scaling Spot-Instances in Ihrem Namen starten kann.
- `elasticloadbalancing`— Registrieren und deregistrieren Sie Instances mit Elastic Load Balancing und überprüfen Sie den Zustand registrierter Ziele.
- `cloudwatch`— CloudWatch Alarmlisten für Skalierungsrichtlinien erstellen, beschreiben, ändern und löschen und Metriken abrufen, die für die prädiktive Skalierung verwendet werden.
- `sns`— Veröffentlichen Sie Benachrichtigungen auf Amazon SNS, wenn Instances gestartet oder beendet werden.
- `events`— EventBridge Regeln in Ihrem Namen erstellen, beschreiben, aktualisieren und löschen.
- `ssm`— Liest Parameter aus dem Parameterspeicher, wenn Sie einen Systems Manager Manager-Parameter als Alias für eine AMI-ID in einer Startvorlage verwenden.
- `vpc-lattice`— Registrieren und deregistrieren Sie Instances bei VPC Lattice und überprüfen Sie den Zustand der registrierten Ziele.
- `resource-groups`— Ruft alle Ressourcennamen (ARNs) der Ressourcen ab, die Mitglieder einer angegebenen Ressourcengruppe sind.

Unterstützte Regionen für Amazon EC2 Auto Scaling serviceverknüpfte Rollen

Amazon EC2 Auto Scaling unterstützt die Verwendung von serviceverknüpften Rollen überall AWS-Regionen dort, wo der Service verfügbar ist.

Eine serviceverknüpfte Rolle erstellen, bearbeiten und löschen

Erstellen einer serviceverknüpften Rolle (automatisch)

Amazon EC2 Auto Scaling erstellt die serviceverknüpfte Rolle `AWSServiceRoleForAutoScaling` für Sie, wenn Sie das erste Mal eine Auto Scaling-Gruppe erstellen, es sei denn, Sie erstellen manuell eine serviceverknüpfte Rolle mit benutzerdefiniertem Suffix und geben diese beim Erstellen der Gruppe an.

Sie müssen über IAM-Berechtigungen zum Erstellen der serviceverknüpften Rolle verfügen. Andernfalls schlägt das automatische Erstellen fehl. Weitere Informationen finden Sie unter [Berechtigungen von serviceverknüpften Rollen](#) im IAM-Benutzerhandbuch und unter [Erstellen einer serviceverknüpften Rolle](#) in diesem Handbuch.

Erstellen einer serviceverknüpften Rolle (manuell)

So erstellen Sie eine serviceverknüpfte Rolle (Konsole)

1. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und Create Role (Rolle erstellen) aus.
3. Wählen Sie für Select trusted entity (Vertrauenswürdige Entität auswählen) die Option AWS - Dienst.
4. Wählen Sie bei Choose the service that will use this role (Service auswählen, der diese Rolle verwendet) die Option EC2 Auto Scaling und den Anwendungsfall EC2 Auto Scaling aus.
5. Wählen Sie Next: Permissions (Nächster Schritt: Berechtigungen), Next: Tags (Nächster Schritt: Tags) und dann Next: Review (Nächster Schritt: Prüfen) aus. Hinweis: Während der Erstellung können keine Tags an serviceverknüpfte Rollen angefügt werden.
6. Lassen Sie auf der Seite „Überprüfen“ das Feld Rollename leer, um eine dienstbezogene Rolle mit dem Namen zu erstellen `AWSServiceRoleForAutoScaling`, oder geben Sie ein Suffix ein, um eine dienstbezogene Rolle mit dem Namen `_` zu erstellen. `AWSServiceRoleForAutoScaling` ***suffix***
7. (Optional:) Bearbeiten Sie in Role description (Rollenbeschreibung) die Beschreibung für die neue serviceverknüpfte Rolle.
8. Wählen Sie Rolle erstellen aus.

So erstellen Sie eine serviceverknüpfte Rolle (AWS CLI)

Verwenden Sie den folgenden [create-service-linked-role](#) CLI-Befehl, um eine serviceverknüpfte Rolle für Amazon EC2 Auto Scaling mit dem Namen `AWSServiceRoleForAutoScaling_` zu erstellen.

suffix

```
aws iam create-service-linked-role --aws-service-name autoscaling.amazonaws.com --  
custom-suffix suffix
```

Die Ausgabe dieses Befehls umfasst den ARN der serviceverknüpften Rolle, den Sie verwenden können, um der serviceverknüpften Rolle Zugriff auf Ihren vom Kunden verwalteten Schlüssel zu erteilen.

```
{  
  "Role": {  
    "RoleId": "ABCDEF0123456789ABCDEF",  
    "CreateDate": "2018-08-30T21:59:18Z",  
    "RoleName": "AWSServiceRoleForAutoScaling_suffix",  
    "Arn": "arn:aws:iam::123456789012:role/aws-service-role/  
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_suffix",  
    "Path": "/aws-service-role/autoscaling.amazonaws.com/",  
    "AssumeRolePolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Action": [  
            "sts:AssumeRole"  
          ],  
          "Principal": {  
            "Service": [  
              "autoscaling.amazonaws.com"  
            ]  
          },  
          "Effect": "Allow"  
        }  
      ]  
    }  
  }  
}
```

Weitere Informationen finden Sie unter [Erstellen einer serviceverknüpften Rolle](#) im IAM-Benutzerhandbuch.

Bearbeiten der serviceverknüpften Rolle

Die für Amazon EC2 Auto Scaling erstellten serviceverknüpften Rollen können nicht bearbeitet werden. Nach dem Erstellen einer serviceverknüpften Rolle können Sie weder den Namen der Rolle noch ihre Berechtigungen ändern. Sie können jedoch die Beschreibung der Rolle bearbeiten. Weitere Informationen finden Sie unter [Bearbeiten einer serviceverknüpften Rollenbeschreibung](#) im IAM-Benutzerhandbuch.

Löschen der serviceverknüpften Rolle

Wenn Sie eine Auto Scaling-Gruppe nicht verwenden, empfehlen wir, deren serviceverknüpfte Rolle zu löschen. Das Löschen der Rolle verhindert, dass Sie eine Entität haben, die nicht verwendet oder aktiv überwacht und verwaltet wird.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem die zugehörigen abhängigen Ressourcen gelöscht wurden. Dies schützt Sie vor der versehentlichen Aufkündigung von Amazon EC2 Auto Scaling-Berechtigungen für Ihre Ressourcen. Wenn eine serviceverknüpfte Rolle mit mehreren Auto Scaling-Gruppen verwendet wird, müssen Sie zunächst alle Auto Scaling-Gruppen, welche die serviceverknüpfte Rolle verwenden, löschen, bevor Sie sie löschen können. Weitere Informationen finden Sie unter [Löschen der Auto-Scaling-Infrastruktur](#).

Sie können IAM zum Löschen der serviceverknüpften Rolle verwenden. Weitere Informationen finden Sie unter [Löschen einer serviceverknüpften Rolle](#) im IAM-Benutzerhandbuch.

Falls Sie die serviceverknüpfte Rolle `AWSServiceRoleForAutoScaling` gelöscht haben, erstellt Amazon EC2 Auto Scaling die Rolle wieder, wenn Sie eine Auto Scaling-Gruppe erstellen und keine andere serviceverknüpfte Rolle angeben.

Beispiele für identitätsbasierte Amazon EC2 Auto Scaling-Richtlinien

Standardmäßig AWS-Konto hat ein brandneuer Benutzer in Ihrem Bereich keine Rechte, etwas zu tun. Ein IAM-Administrator muss IAM-Richtlinien erstellen und zuweisen, die einer IAM-Identität (etwa einem Benutzer oder einer Rolle) die Berechtigung zum Ausführen von API-Aktionen von Amazon EC2 Auto Scaling gewähren.

Informationen dazu, wie Sie unter Verwendung dieser Beispiel-JSON-Richtliniendokumente eine IAM-Richtlinie erstellen, finden Sie unter [Erstellen von Richtlinien auf der JSON-Registerkarte](#) im IAM-Benutzerhandbuch.

Dies ist ein Beispiel für eine Berechtigungsrichtlinie.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling>DeleteAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
      }
    },
    {
      "Effect": "Allow",
      "Action": "autoscaling:Describe*",
      "Resource": "*"
    }
  ]
}
```

Diese Beispielrichtlinie gewährt Benutzern Berechtigungen zum Erstellen, Aktualisieren und Löschen von Auto-Scaling-Gruppen, jedoch nur, wenn die Gruppe das Tag **purpose=testing** verwendet. Da Describe-Aktionen keine Berechtigungen auf Ressourcenebene unterstützen, müssen Sie sie in einer separaten Anweisung ohne Bedingungen angeben. Um Instances mit einer Startvorlage zu starten, muss der Benutzer auch über die `ec2:RunInstances`-Berechtigung verfügen. Weitere Informationen finden Sie unter [Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling-Gruppen](#).

Note

Sie können auch eigene benutzerdefinierte IAM-Richtlinien erstellen, um IAM-Identitäten (Benutzern oder Rollen) Berechtigungen zum Ausführen von Amazon-EC2-Auto-Scaling-Aktionen zu gewähren oder zu verweigern. Die benutzerdefinierten Richtlinien können Sie dann den IAM-Identitäten zuweisen, die die angegebenen Berechtigungen fordern. Die folgenden Beispiele zeigen Berechtigungen für einige häufige Anwendungsfälle.

Bei einigen Amazon EC2 Auto Scaling-API-Aktionen lassen sich bestimmte Auto Scaling-Gruppen, die mit der Aktion erstellt oder geändert werden können, in die Richtlinie einbinden. Sie können die Zielressourcen für diese Aktionen einschränken, indem Sie eine einzelne Auto Scaling Scaling-Gruppe angeben ARNs. Als bewährte Methode empfehlen wir jedoch, tagbasierte Richtlinien zu verwenden, die Aktionen für Auto Scaling-Gruppen mit einem bestimmten Tag zulassen (oder ablehnen).

Beispiele

- [Steuern Sie die Größe der Auto-Scaling-Gruppen, die erstellt werden können](#)
- [Steuern, welche Tag-Schlüssel und Tag-Werte verwendet werden können](#)
- [Steuern Sie, welche Auto-Scaling-Gruppen gelöscht werden können](#)
- [Steuern, welche Skalierungsrichtlinien gelöscht werden können](#)
- [Steuern Sie den Zugriff auf Aktionen zur Instance-Aktualisierung](#)
- [Erstellen einer serviceverknüpften Rolle](#)
- [Steuern Sie, welche serviceverknüpfte Rolle übergeben werden kann \(mit\) PassRole](#)

Steuern Sie die Größe der Auto-Scaling-Gruppen, die erstellt werden können

Die folgende Richtlinie gewährt Berechtigungen zum Erstellen und Aktualisieren aller Auto-Scaling-Gruppen mit dem Tag **environment=development**, sofern der Anforderer nicht eine Mindestgröße kleiner als **1** oder eine maximale Größe größer als **10** angibt. Verwenden Sie nach Möglichkeit Tags, um den Zugriff auf die Auto-Scaling-Gruppen in Ihrem Konto zu steuern.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:UpdateAutoScalingGroup"
    ],
    "Resource": "*",
    "Condition": {
```

```

    "StringEquals": { "autoscaling:ResourceTag/environment":
      "development" },
    "NumericGreaterThanEqualsIfExists": { "autoscaling:MinSize": 1 },
    "NumericLessThanEqualsIfExists": { "autoscaling:MaxSize": 10 }
  }
}]
}

```

Wenn Sie keine Tags verwenden, um den Zugriff auf Auto Scaling Scaling-Gruppen zu steuern, können Sie diese auch verwenden, ARNs um die Auto Scaling Scaling-Gruppen zu identifizieren, für die die IAM-Richtlinie gilt.

Eine Auto Scaling-Gruppe hat den folgenden ARN.

```

"Resource": "arn:aws:autoscaling:region:account-id:autoScalingGroup:*:autoScalingGroupName/my-asg"

```

Sie können auch mehrere ARNs angeben, indem Sie sie in eine Liste einschließen. Weitere Informationen zum Angeben der ARNs von Amazon EC2 Auto-Scaling-Ressourcen im Resource-Element finden Sie unter [Richtlinienressourcen für Amazon EC2 Auto Scaling](#).

Steuern, welche Tag-Schlüssel und Tag-Werte verwendet werden können

Sie können auch Bedingungen in Ihren IAM-Richtlinien verwenden, um die Tag-Schlüssel und Tag-Werte zu steuern, die auf Auto-Scaling-Gruppen angewendet werden können. Verwenden Sie den `aws:RequestTag`-Bedingungsschlüssel, um Berechtigungen zum Erstellen oder Markieren einer Auto-Scaling-Gruppe nur dann zu gewähren, wenn der Anforderer bestimmte Tags angibt. Um nur bestimmte Tag-Schlüssel zuzulassen, verwenden Sie den Bedingungsschlüssel `aws:TagKeys` mit dem Modifikator `ForAllValues`.

Die folgende Richtlinie erfordert, dass der Anforderer in der Anfrage ein Tag mit dem Schlüssel **environment** angibt. Der Wert `"?*"` erzwingt, dass ein Wert für den Tag-Schlüssel vorhanden ist. Bei der Verwenden eines Platzhalters müssen Sie den `StringLike`-Bedingungsoperator verwenden.

JSON

```
{
```

```

"Version":"2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "autoscaling:CreateAutoScalingGroup",
    "autoscaling:CreateOrUpdateTags"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": { "aws:RequestTag/environment": "?*" }
  }
}]
}

```

Die folgende Richtlinie legt fest, dass der Antragssteller nur Auto-Scaling-Gruppen mit den Tags **purpose=webserver** und **cost-center=cc123** kennzeichnen kann und nur die Tags **purpose** und **cost-center** zulässt (keine anderen Tags können angegeben werden).

JSON

```


{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:CreateOrUpdateTags"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/purpose": "webserver",
        "aws:RequestTag/cost-center": "cc123"
      },
      "ForAllValues:StringEquals": { "aws:TagKeys": [purpose, "cost-
center"] }
    }
  }]
}

```

Die folgende Richtlinie erfordert, dass der Anforderer mindestens ein Tag in der Anfrage angibt, und lässt nur die **cost-center**- und **owner**-Schlüssel zu.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:CreateOrUpdateTags"
    ],
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": { "aws:TagKeys": ["cost-center", "owner] }
    }
  }]
}
```

 Note

Bei Bedingungen gilt, dass die Groß- und Kleinschreibung für den Bedingungsschlüssel nicht berücksichtigt und für den Bedingungswert beachtet wird. Verwenden Sie aus diesem Grund den `aws:TagKeys`-Bedingungsschlüssel und geben Sie den Tag (Markierung)-Schlüssel als Wert dieser Bedingung an, wenn Sie die Berücksichtigung der Groß- und Kleinschreibung für einen Tag (Markierung)-Schlüssel erzwingen möchten.

Steuern Sie, welche Auto-Scaling-Gruppen gelöscht werden können

Die folgende Richtlinie erlaubt das Löschen einer Auto-Scaling-Gruppe nur, wenn die Gruppe über das Tag verfügt **environment=development**.

JSON

```
{
  "Version":"2012-10-17",
```

```

"Statement": [{
  "Effect": "Allow",
  "Action": "autoscaling:DeleteAutoScalingGroup",
  "Resource": "*",
  "Condition": {
    "StringEquals": { "aws:ResourceTag/environment": "development" }
  }
}]
}

```

Die folgende Richtlinie verwendet den `autoscaling:ForceDelete` Bedingungsschlüssel, um den Zugriff auf die `DeleteAutoScalingGroup` API-Aktion zu steuern. Dadurch kann verhindert werden, dass bestimmte Benutzer den Vorgang zum erzwungenen Löschen verwenden, bei dem beim Löschen der Gruppe alle Instances innerhalb einer Auto Scaling Scaling-Gruppe beendet werden.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": "autoscaling:DeleteAutoScalingGroup",
    "Resource": "*",
    "Condition": {
      "Bool": {
        "autoscaling:ForceDelete": "true"
      }
    }
  }]
}

```

Wenn Sie keine Bedingungsschlüssel verwenden, um den Zugriff auf Auto Scaling Scaling-Gruppen zu steuern, können Sie stattdessen die Anzahl ARNs der Ressourcen in dem Resource Element angeben, um den Zugriff zu steuern.

Die folgende Richtlinie erteilt Benutzern die Erlaubnis, die `DeleteAutoScalingGroup` API-Aktion zu verwenden, jedoch nur für Auto Scaling Scaling-Gruppen, deren Name mit `beginntdevteam-`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "arn:aws:autoscaling:us-east-1:111122223333:autoScalingGroup:*:autoScalingGroupName/devteam-*"
    }
  ]
}
```

Sie können auch mehrere angeben, ARNs indem Sie sie in eine Liste aufnehmen. Einbeziehung der UUID stellt sicher, dass Zugriff zu der spezifischen Auto Scaling-Gruppe gewährt ist. Die UUID für eine neue Gruppe unterscheidet sich von der UUID für eine gelöschte Gruppe mit demselben Namen.

```
"Resource": [
  "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-1",
  "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-2",
  "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-3"
]
```

Steuern, welche Skalierungsrichtlinien gelöscht werden können

Die folgende Richtlinie gewährt Berechtigungen zum Verwenden der DeletePolicy-Aktion zum Löschen einer Skalierungsrichtlinie. Es lehnt die Aktion jedoch auch dann ab, wenn die Auto-Scaling-Gruppe, auf welche die Aktion abzielt, über das Tag **environment=production** verfügt. Verwenden Sie nach Möglichkeit Tags, um den Zugriff auf die Auto-Scaling-Gruppen in Ihrem Konto zu steuern.

JSON

```
{
```

```

"Version":"2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Action": "autoscaling:DeletePolicy",
  "Resource": "*"
},
{
  "Effect": "Deny",
  "Action": "autoscaling:DeletePolicy",
  "Resource": "*",
  "Condition": {
    "StringEquals": { "autoscaling:ResourceTag/environment": "production" }
  }
}]
}

```

Steuern Sie den Zugriff auf Aktionen zur Instance-Aktualisierung

Die folgende Richtlinie erteilt nur dann Berechtigungen zum Starten, Zurücksetzen und Abbrechen einer Instance-Aktualisierung, wenn die Auto-Scaling-Gruppe, auf welche die Aktion abzielt, über das Tag **environment=testing** verfügt. Da Describe-Aktionen keine Berechtigungen auf Ressourcenebene unterstützen, müssen Sie sie in einer separaten Anweisung ohne Bedingungen angeben.

JSON

```

{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:StartInstanceRefresh",
      "autoscaling:CancelInstanceRefresh",
      "autoscaling:RollbackInstanceRefresh"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": { "autoscaling:ResourceTag/environment": "testing" }
    }
  },
  {

```

```
    "Effect": "Allow",
    "Action": "autoscaling:DescribeInstanceRefreshes",
    "Resource": "*"
  }
}
```

Um eine gewünschte Konfiguration im `StartInstanceRefresh`-Aufruf anzugeben, benötigen Benutzer möglicherweise einige zugehörige Berechtigungen, wie beispielsweise:

- `ec2:RunInstances` — Um EC2-Instances mithilfe einer Startvorlage zu starten, muss der Benutzer über die `ec2:RunInstances` entsprechende Berechtigung in einer IAM-Richtlinie verfügen. Weitere Informationen finden Sie unter [Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling-Gruppen](#).
- `ec2:CreateTags` — Um EC2-Instances von einer Startvorlage aus zu starten, die den Instances und Volumes bei der Erstellung Tags hinzufügt, muss der Benutzer über die `ec2:CreateTags` entsprechende Berechtigung in einer IAM-Richtlinie verfügen. Weitere Informationen finden Sie unter [Erforderliche Berechtigungen zum Markieren von Instances und Volumes](#).
- `iam:PassRole` — Um EC2-Instances von einer Startvorlage aus zu starten, die ein Instance-Profil (einen Container für eine IAM-Rolle) enthält, muss der Benutzer auch über die `iam:PassRole` entsprechende Berechtigung in einer IAM-Richtlinie verfügen. Weitere Informationen und eine IAM-Beispielrichtlinie finden Sie unter [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#).
- `ssm:GetParameters` — Um EC2-Instances von einer Startvorlage aus zu starten, die einen AWS Systems Manager Parameter verwendet, muss der Benutzer auch über die `ssm:GetParameters` entsprechende Berechtigung in einer IAM-Richtlinie verfügen. Weitere Informationen finden Sie unter [Verwenden Sie AWS Systems Manager Parameter anstelle von AMI IDs in Startvorlagen](#).

Erstellen einer serviceverknüpften Rolle

Amazon EC2 Auto Scaling benötigt Berechtigungen zum Erstellen einer serviceverknüpften Rolle, wenn ein Benutzer in Ihnen zum ersten Mal Amazon EC2 Auto Scaling Scaling-API-Aktionen AWS-Konto aufruft. Wenn die serviceverknüpfte Rolle noch nicht vorhanden ist, erstellt Amazon EC2 Auto Scaling diese in Ihrem Konto. Die serviceverknüpfte Rolle erteilt Amazon EC2 Auto Scaling Berechtigungen, sodass Amazon EC2 Auto Scaling andere in AWS-Services Ihrem Namen anrufen kann.

Damit diese automatische Rollenerstellung möglich ist, müssen Benutzer über Berechtigungen für die Aktion `iam:CreateServiceLinkedRole` verfügen.

```
"Action": "iam:CreateServiceLinkedRole"
```

Im Folgenden wird ein Beispiel für eine Berechtigungsrichtlinie gezeigt, mit der ein Benutzer eine Amazon EC2 Auto Scaling-Rolle für Amazon EC2 Auto Scaling erstellen kann.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "autoscaling.amazonaws.com" }
      }
    }
  ]
}
```

Steuern Sie, welche serviceverknüpfte Rolle übergeben werden kann (mit `PassRole`

Benutzer, die Auto-Scaling-Gruppen erstellen oder aktualisieren und in der Anfrage eine serviceverknüpfte Rolle mit benutzerdefiniertem Suffix angeben, benötigen die `iam:PassRole`-Berechtigung.

Sie können die `iam:PassRole` Berechtigung verwenden, um die Sicherheit Ihrer vom AWS KMS Kunden verwalteten Schlüssel zu schützen, indem Sie verschiedenen dienstbezogenen Rollen Zugriff auf verschiedene Schlüssel gewähren. Abhängig von den Anforderungen Ihrer Organisation haben Sie möglicherweise einen Schlüssel für das Entwicklungsteam, einen weiteren für das QA-Team und einen weiteren für das Finanzteam. Erstellen Sie zunächst eine serviceverknüpfte Rolle, die auf den erforderlichen Schlüssel zugreifen kann, z. B. eine serviceverknüpfte Rolle mit dem Namen `AWSServiceRoleForAutoScaling_devteamkeyaccess`. Fügen Sie dann diese Richtlinie an eine IAM-Identität an, z. B. einen Benutzer oder eine Rolle.

Die folgende Richtlinie gewährt die Berechtigung, die **AWSServiceRoleForAutoScaling_devteamkeyaccess** Rolle an jede Auto-Scaling-Gruppe weiterzugeben, deren Name mit **devteam-** beginnt. Wenn die IAM-Identität, die die Auto-Scaling-Gruppe erstellt, versucht, eine andere serviceverknüpfte Rolle anzugeben, wird eine Fehlermeldung ausgegeben. Wenn sie keine serviceverknüpfte Rolle angeben, wird stattdessen die standardmäßige **AWSServiceRoleForAutoScaling** Rolle verwendet.

Weitere Informationen zu serviceverknüpfte Rollen mit benutzerdefiniertem Suffix finden Sie unter [Serviceverknüpfte Rollen für Amazon EC2 Auto Scaling](#).

Serviceübergreifende Confused-Deputy-Prävention

Das Problem des verwirrten Stellvertreters ist ein Sicherheitsproblem, bei dem eine Entität, die keine Berechtigung zur Durchführung einer Aktion hat, eine privilegiertere Entität zur Durchführung der Aktion zwingen kann.

Ein AWS dienstübergreifendes Identitätswechsels kann zu einem Problem mit dem verwirrten Stellvertreter führen. Ein dienstübergreifender Identitätswechsel kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der Anruf-Dienst kann so manipuliert werden, dass er seine Berechtigungen verwendet, um auf die Ressourcen eines anderen Kunden zu reagieren, auf die er sonst nicht zugreifen dürfte.

Um dies zu verhindern, AWS bietet Tools, mit denen Sie Ihre Daten für alle Dienste mit Dienstprinzipalen schützen können, denen Zugriff auf Ressourcen in Ihrem Konto gewährt wurde. Wir empfehlen den Einsatz des [aws:SourceArn](#) und [aws:SourceAccount](#) globale Bedingungskontext-Schlüssel in Vertrauensrichtlinien für Amazon EC2 Auto Scaling Servicerollen. Diese Schlüssel beschränken die Berechtigungen, die Amazon EC2 Auto Scaling der Ressource einem anderen Service gewährt.

Die Werte für die `SourceAccount` Felder `SourceArn` und werden festgelegt, wenn Amazon EC2 Auto Scaling AWS -Security-Token-Service (AWS STS) verwendet, um eine Rolle in Ihrem Namen zu übernehmen.

Um die globalen Bedingungschlüssel `aws:SourceArn` oder `aws:SourceAccount` zu verwenden, legen Sie den Wert auf den Amazon-Ressourcennamen (ARN) oder das Konto der Ressource, die Amazon EC2 Auto Scaling speichert. Nutzen Sie, wann immer möglich, den spezifischeren Wert `aws:SourceArn`. Legen Sie den Wert auf den ARN oder ein ARN-Muster mit Platzhalterzeichen fest (*) für die unbekannt Teile des ARN. Wenn Sie den ARN der Ressource nicht kennen, verwenden Sie stattdessen `aws:SourceAccount`.

Das folgende Beispiel zeigt, wie Sie die globalen Bedingungskontextschlüssel `aws:SourceArn` und `aws:SourceAccount` in Amazon EC2 Auto Scaling verwenden können, um das Problem des verwirrten Stellvertreters zu vermeiden.

Beispiel: Verwenden `aws:SourceArn` und `aws:SourceAccount`-Bedingungsschlüssel

Eine Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen durchzuführen, wird als [Servicerolle](#) bezeichnet. In Fällen, in denen Sie Lifecycle-Hooks erstellen möchten, die Benachrichtigungen an einen anderen Ort als Amazon senden EventBridge, müssen Sie eine Servicerolle erstellen, damit Amazon EC2 Auto Scaling in Ihrem Namen Benachrichtigungen an ein Amazon SNS SNS-Thema oder eine Amazon SQS SQS-Warteschlange senden kann. Wenn Sie nur eine Auto Scaling-Gruppe mit dem betriebsübergreifenden Zugriff verknüpfen möchten, können Sie die Vertrauensrichtlinie der Servicerolle wie folgt angeben.

In dieser Beispiel-Vertrauensrichtlinie werden Bedingungsanweisungen verwendet, um die `AssumeRole`-Fähigkeit für die Servicerolle nur für die Aktionen, die sich auf die angegebene Auto Scaling-Gruppe im angegebenen Konto auswirken. Die Bedingungen `aws:SourceArn` und `aws:SourceAccount` werden unabhängig ausgewertet. Jede Anforderung, die die Servicerolle zu verwenden, muss beide Bedingungen erfüllen.

Bevor Sie diese Schlüsselrichtlinie verwenden, ersetzen Sie die Beispiel-Konto-ID, die Region und den Trail-Namen durch gültige Werte aus Ihrem Konto.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConfusedDeputyPreventionExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "autoscaling.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:autoscaling:us-east-1:111122223333:autoScalingGroup:uuid:autoScalingGroupName/my-asg"
        },
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

```
}  
  }  
} }
```

Für das obige Beispiel gilt:

- Das `Principal`-Element gibt den Auftraggeber des Dienstes an (`autoscaling.amazonaws.com`).
- Das `Action`-Element spezifiziert die `sts:AssumeRole` Aktion.
- Das `Condition`-Element spezifiziert die globalen Bedingungsschlüssel `aws:SourceArn` und `aws:SourceAccount`. Der ARN der Quelle enthält die Konto-ID, daher ist es nicht erforderlich, `aws:SourceAccount` mit `aws:SourceArn` zu verwenden.

Zusätzliche Informationen

Weitere Informationen finden Sie unter [AWS Global Condition Context Keys](#), [The confused deputy problem](#) und [Update a role trust policy](#) im IAM-Benutzerhandbuch.

Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling-Gruppen

Amazon EC2 Auto Scaling unterstützt die Verwendung von Amazon EC2-Startvorlagen mit Ihren Auto Scaling-Gruppen. Es wird empfohlen, Benutzern das Erstellen von Auto Scaling-Gruppen anhand von Startvorlagen zu gestatten, da sie auf diese Weise die neuesten Funktionen von Amazon EC2 Auto Scaling und Amazon EC2 verwenden können. Beispielsweise müssen Benutzer eine Startvorlage angeben, um eine [Richtlinie für gemischte Instances](#) verwenden zu können.

Sie können die `AmazonEC2FullAccess`-Richtlinie verwenden, um Benutzern Vollzugriff für die Arbeit mit Amazon EC2 Auto Scaling-Ressourcen, Startvorlagen und anderen EC2-Ressourcen in ihrem Konto zu gewähren. Sie können auch eigene benutzerdefinierte IAM-Richtlinien erstellen, um Benutzern detaillierte Berechtigungen zum Arbeiten mit Startvorlagen zu erteilen, wie in diesem Thema beschrieben.

Eine Beispielrichtlinie, die Sie für Ihre eigene Verwendung anpassen können

Das folgende Beispiel zeigt eine grundlegende Berechtigungsrichtlinie, die Sie für Ihre eigene Verwendung anpassen können. Die Richtlinie gewährt Berechtigungen zum Erstellen, Aktualisieren

und Löschen aller Auto-Scaling-Gruppen, jedoch nur, wenn die Gruppe das Tag **purpose=testing** verwendet. Diese gewährt dann Berechtigung für alle Describe-Aktionen. Da Describe-Aktionen keine Berechtigungen auf Ressourcenebene unterstützen, müssen Sie sie in einer separaten Anweisung ohne Bedingungen angeben.

IAM-Identitäten (Benutzer oder Rollen) mit dieser Richtlinie verfügen über die Berechtigung zum Erstellen oder Aktualisieren einer Auto-Scaling-Gruppe mithilfe einer Startvorlage, da sie auch über die Berechtigung zur Verwendung der `ec2:RunInstances`-Aktion verfügen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling>DeleteAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:Describe*",
        "ec2:RunInstances"
      ],
      "Resource": "*"
    }
  ]
}
```

Benutzer, die Auto-Scaling-Gruppen erstellen oder aktualisieren, benötigen möglicherweise einige zugehörige Berechtigungen, wie beispielsweise:

- `ec2:CreateTags` — Um den Instances und Volumes bei der Erstellung Tags hinzuzufügen, muss der Benutzer über die `ec2:CreateTags` entsprechende Berechtigung in einer IAM-Richtlinie verfügen. Weitere Informationen finden Sie unter [Erforderliche Berechtigungen zum Markieren von Instances und Volumes](#).
- `iam:PassRole` — Um EC2-Instances von einer Startvorlage aus zu starten, die ein Instance-Profil (einen Container für eine IAM-Rolle) enthält, muss der Benutzer auch über die `iam:PassRole` entsprechende Berechtigung in einer IAM-Richtlinie verfügen. Weitere Informationen und eine IAM-Beispielrichtlinie finden Sie unter [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#).
- `ssm:GetParameters` — Um EC2-Instances von einer Startvorlage aus zu starten, die einen AWS Systems Manager Parameter verwendet, muss der Benutzer auch über die `ssm:GetParameters` entsprechende Berechtigung in einer IAM-Richtlinie verfügen. Weitere Informationen finden Sie unter [Verwenden Sie AWS Systems Manager Parameter anstelle von AMI IDs in Startvorlagen](#).

Diese Berechtigungen für Aktionen, die beim Starten von Instances ausgeführt werden sollen, werden überprüft, wenn der Benutzer mit einer Auto-Scaling-Gruppe interagiert. Weitere Informationen finden Sie unter [Überprüfung der Berechtigungen für `ec2:RunInstances` und `iam:PassRole`](#).

Die folgenden Beispiele zeigen Richtlinienanweisungen, die Sie verwenden können, um den Zugriff von IAM-Benutzern auf Startvorlagen zu steuern.

Themen

- [Verlangen, dass Startvorlagen ein bestimmtes Tag haben](#)
- [Eine Startvorlage und eine Versionsnummer verlangen](#)
- [Erfordert die Verwendung des Instance-Metadatendienstes Version 2 \(\) IMDSv2](#)
- [Beschränken des Zugriffs auf Amazon EC2-Ressourcen](#)
- [Erforderliche Berechtigungen zum Markieren von Instances und Volumes](#)
- [Zusätzliche Berechtigungen für Startvorlagen](#)
- [Überprüfung der Berechtigungen für `ec2:RunInstances` und `iam:PassRole`](#)
- [Zugehörige Ressourcen](#)

Verlangen, dass Startvorlagen ein bestimmtes Tag haben

Bei der Erteilung von `ec2:RunInstances` Berechtigungen können Sie angeben, dass Benutzer beim Starten von Instances mit einer Startvorlage nur Startvorlagen mit bestimmten Tags oder

spezifischen IDs Zugriffsbeschränkungen verwenden können. Sie können auch das AMI und andere Ressourcen steuern, auf die jeder, der Startvorlagen verwendet, beim Starten von Instances verweisen und diese verwenden kann, indem Sie zusätzliche Berechtigungen auf Ressourcenebene für den `RunInstances`-Aufruf angeben.

Das folgende Beispiel schränkt die Berechtigungen für die Aktion `ec2:RunInstances` auf das Starten von Vorlagen ein, die sich in der angegebenen Region befinden und die das Tag **purpose=testing** haben. Außerdem erhalten Benutzer Zugriff auf die in einer Startvorlage angegebenen Ressourcen: Instance-Typen AMIs, Volumes, Schlüsselpaare, Netzwerkschnittstellen und Sicherheitsgruppen.

Weitere Informationen zur Verwendung von tagbasierten Richtlinien mit Startvorlagen finden Sie unter [Steuern des Zugriffs auf Startvorlagen mit IAM-Berechtigungen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Eine Startvorlage und eine Versionsnummer verlangen

Sie können IAM-Berechtigungen auch verwenden, um zu erzwingen, dass beim Erstellen oder Aktualisieren von Auto-Scaling-Gruppen eine Startvorlage und die Versionsnummer der Startvorlage angegeben werden müssen.

Im folgenden Beispiel können Benutzer Auto-Scaling-Gruppen nur erstellen und aktualisieren, wenn eine Startvorlage und die Versionsnummer der Startvorlage angegeben sind. Wenn Benutzer mit dieser Richtlinie die Versionsnummer weglassen, um entweder die Version der Startvorlage `$Latest` oder `$Default` anzugeben, oder versuchen, stattdessen eine Startkonfiguration zu verwenden, schlägt die Aktion fehl.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
```

```

        "Bool": { "autoscaling:LaunchTemplateVersionSpecified": "true" }
    },
    {
        "Effect": "Deny",
        "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:UpdateAutoScalingGroup"
        ],
        "Resource": "*",
        "Condition": {
            "Null": { "autoscaling:LaunchConfigurationName": "false" }
        }
    }
]
}

```

Erfordert die Verwendung des Instance-Metadatendienstes Version 2 () IMDSv2

Für zusätzliche Sicherheit können Sie die Berechtigungen Ihrer Benutzer so einrichten, dass sie die Verwendung einer Startvorlage erfordern, die Folgendes erfordert IMDSv2. Weitere Informationen finden Sie unter [Konfiguration des Instance-Metadaten-Service](#) im Amazon EC2 EC2-Benutzerhandbuch.

Das folgende Beispiel legt fest, dass Benutzer die `ec2:RunInstances` Aktion nur aufrufen können, wenn für die Instance auch die Verwendung von aktiviert wurde IMDSv2 (angegeben durch `"ec2:MetadataHttpTokens": "required"`).

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireImdsV2",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "StringNotEquals": { "ec2:MetadataHttpTokens": "required" }
      }
    }
  ]
}

```

```
    }  
  ]  
}
```

Tip

Um zu erzwingen, dass Ersatz-Instances von Auto Scaling gestartet werden, die eine neue Startvorlage oder eine neue Version einer Startvorlage mit den konfigurierten Instance-Metadatenoptionen verwenden, können Sie eine Instance-Aktualisierung starten. Weitere Informationen finden Sie unter [Aktualisieren von Auto-Scaling-Instances](#).

Beschränken des Zugriffs auf Amazon EC2-Ressourcen

Die folgenden Beispiele zeigen, wie Sie Berechtigungen auf Ressourcenebene und markierte Ressourcen verwenden, um den Zugriff auf Amazon EC2 EC2-Ressourcen einzuschränken.

Beispiel 1: Beschränken Sie den Start von Amazon EC2 EC2-Instances auf bestimmte Ressourcen und Instance-Typen

Das folgende Beispiel steuert die Konfiguration der Instances, die ein Benutzer starten kann, indem der Zugriff auf Amazon EC2-Ressourcen eingeschränkt wird. Um Berechtigungen auf Ressourcenebene für Ressourcen festzulegen, die in einer Startvorlage angegeben sind, müssen Sie die Ressourcen in die RunInstances-Aktionsanweisung aufnehmen.

In diesem Beispiel gibt es zwei Anweisungen:

- Die erste Anweisung erfordert, dass Benutzer Instances in einem bestimmten Subnetz (**subnet-1a2b3c4d**) starten und dabei eine bestimmte Sicherheitsgruppe (**sg-903004f88example**) und ein bestimmtes AMI (**ami-04d5cc9b88example**) verwenden. Außerdem erhalten die Benutzer Zugriff auf die in einer Startvorlage angegebenen Ressourcen: Netzwerkschnittstellen, Schlüsselpaare und Volumes.
- Die zweite Anweisung ermöglicht es den Benutzern, Instances nur mit den Instance-Typen **t2.micro** und **t2.small** zu starten, was aus Kostengründen sinnvoll ist.

Beachten Sie jedoch, dass es derzeit keine effektive Methode gibt, um Benutzer, die berechtigt sind, Instances mit einer Startvorlage zu starten, vollständig daran zu hindern, andere Instance-Typen zu starten. Dies liegt daran, dass ein in einer Startvorlage angegebener Instance-

Typ überschrieben werden kann, sodass Instance-Typen verwendet werden, die mithilfe der attributbasierten Instance-Typauswahl definiert wurden.

Eine vollständige Liste der Berechtigungen auf Ressourcenebene, mit denen Sie die Konfiguration der Instances steuern können, die ein Benutzer starten kann, finden Sie unter [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EC2](#) in der Referenz der Serviceberechtigung.

Beispiel 2: Für den Start von Amazon EC2 EC2-Instances sind Tags erforderlich und der Ressourcenzugriff eingeschränkt

Die folgende Beispielrichtlinie zeigt, wie Sie Bedingungen in Ihrer identitätsbasierten Richtlinie verwenden können, um den Zugriff auf Amazon EC2 EC2-Ressourcen anhand von Tags zu steuern.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "InstanceTags",
      "Effect": "Allow",
      "Action": [
        "ec2:RunInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/owner": "dev"
        }
      }
    },
    {
      "Sid": "InstanceBoundaries",
      "Effect": "Allow",
      "Action": [
        "ec2:RunInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1::image/*",
```

```

        "arn:aws:ec2:us-east-1:555555555555:subnet/*",
        "arn:aws:ec2:us-east-1:555555555555:network-interface/*"
    ]
},
{
    "Sid": "InstanceResourceTags",
    "Effect": "Allow",
    "Action": [
        "ec2:RunInstances"
    ],
    "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:key-pair/*",
        "arn:aws:ec2:us-east-1:555555555555:security-group/*",
        "arn:aws:ec2:us-east-1:555555555555:launch-template/*",
        "arn:aws:ec2:us-east-1:555555555555:volume/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/purpose": "testing"
        }
    }
}
]
}

```

In diesem Beispiel gibt es drei Aussagen:

- Die erste Anweisung ermöglicht es Benutzern, Instances in der angegebenen Region nur dann zu starten, wenn in der Anfrage ein Tag mit verwendet `owner=dev` wird.
- Die zweite Anweisung gewährt Benutzern Zugriff auf das AMI, das Subnetz und die Netzwerkschnittstelle in der angegebenen Region.
- Die dritte Anweisung ermöglicht es Benutzern, Instances in der angegebenen Region mit einem vorhandenen key pair, einer Sicherheitsgruppe, einer Startvorlage und einem Volume zu starten, die über das Tag verfügen `purpose=testing`.

Weitere Informationen finden Sie unter [Steuern des Zugriffs auf AWS Ressourcen mithilfe von Tags](#) im IAM-Benutzerhandbuch.

Erforderliche Berechtigungen zum Markieren von Instances und Volumes

Das folgende Beispiel ermöglicht es Benutzern, Instances und Volumes bei der Erstellung zu kennzeichnen. Diese Richtlinie wird benötigt, wenn in der Startvorlage Tags angegeben sind. Weitere Informationen finden Sie unter [Erteilen der Berechtigung zum Markieren von Ressourcen während der Erstellung](#) im Amazon-EC2-Benutzerhandbuch.

Zusätzliche Berechtigungen für Startvorlagen

Sie müssen Ihren Konsolenbenutzern Berechtigungen für die `ec2:DescribeLaunchTemplates`- und `ec2:DescribeLaunchTemplateVersions`-Aktionen gewähren. Ohne diese Berechtigungen können Startvorlagendaten nicht im Auto Scaling-Gruppen-Assistenten geladen werden, und Benutzer können den Assistenten nicht durchlaufen, um Instances mithilfe einer Startvorlage zu starten. Sie können diese zusätzlichen Aktionen im `Action`-Element einer IAM-Richtlinienanweisung angeben.

Überprüfung der Berechtigungen für **ec2:RunInstances** und **iam:PassRole**

Benutzer können angeben, welche Version einer Startvorlage ihre Auto-Scaling-Gruppe verwendet. Je nach ihren Berechtigungen kann es sich dabei um eine bestimmte nummerierte Version oder um die Version `$Latest` oder `$Default` der Startvorlage handeln. Wenn Letzteres der Fall ist, seien Sie besonders vorsichtig. Dies kann die Berechtigungen für `ec2:RunInstances` und `iam:PassRole`, die Sie einschränken wollten, außer Kraft setzen.

In diesem Abschnitt wird das Szenario der Verwendung der neuesten oder Standardversion der Startvorlage mit einer Auto-Scaling-Gruppe erläutert.

Wenn ein Benutzer die `CreateAutoScalingGroup`-, `UpdateAutoScalingGroup`- oder `StartInstanceRefresh`-APIs aufruft, prüft Amazon EC2 Auto Scaling seine Berechtigungen anhand der Version der Startvorlage, die zu diesem Zeitpunkt die neueste oder die Standardversion ist, bevor es mit der Anforderung fortfährt. Dadurch werden die Berechtigungen für Aktionen validiert, die beim Starten von Instances abgeschlossen werden müssen, wie z. B. die Aktionen `ec2:RunInstances` und `iam:PassRole`. Um dies zu erreichen, führen wir einen Amazon EC2 [RunInstances](#) EC2-Probelauf durch, um zu überprüfen, ob der Benutzer über die erforderlichen Berechtigungen für die Aktion verfügt, ohne die Anfrage tatsächlich zu stellen. Wenn eine Antwort zurückgegeben wird, wird sie von Amazon EC2 Auto Scaling gelesen. Wenn die Berechtigungen des Benutzers eine bestimmte Aktion nicht zulassen, lässt Amazon EC2 Auto Scaling die Anfrage fehlschlagen und gibt dem Benutzer eine Fehlermeldung mit Informationen über die fehlende Berechtigung zurück.

Nach Abschluss der ersten Überprüfung und Anforderung startet Amazon EC2 Auto Scaling Instances bei jedem Start mit der neuesten Version oder Standardversion, auch wenn sie sich geändert hat. Dabei werden die Berechtigungen der [serviceverknüpften Rolle](#) verwendet. Das bedeutet, dass ein Benutzer, der die Startvorlage verwendet, diese möglicherweise aktualisieren kann, um eine IAM-Rolle an eine Instance zu übergeben, auch wenn er nicht über die `iam:PassRole`-Berechtigung verfügt.

Verwenden Sie den `autoscaling:LaunchTemplateVersionSpecified`-Bedingungsschlüssel, wenn Sie einschränken möchten, wer Zugriff auf die Konfiguration von Gruppen hat, um die Version `$Latest` oder `$Default` zu verwenden. Dadurch wird sichergestellt, dass die Auto Scaling Scaling-Gruppe nur dann eine bestimmte nummerierte Version akzeptiert, wenn ein Benutzer `CreateAutoScalingGroup` und aufruft `UpdateAutoScalingGroup` APIs. Ein Beispiel, das zeigt, wie dieser Bedingungsschlüssel zu einer IAM-Richtlinie hinzugefügt wird, finden Sie unter [Eine Startvorlage und eine Versionsnummer verlangen](#).

Für Auto-Scaling-Gruppen, die für die Verwendung der Startvorlagenversion `$Latest` oder `$Default` konfiguriert sind, sollten Sie einschränken, wer Versionen der Startvorlage erstellen und verwalten kann, einschließlich der `ec2:ModifyLaunchTemplate`-Aktion, die es einem Benutzer ermöglicht, die Standardversion der Startvorlage anzugeben. Weitere Informationen finden Sie unter [Steuern von Versionsberechtigungen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Zugehörige Ressourcen

Weitere Informationen zu Berechtigungen zum Anzeigen, Erstellen und Löschen von Startvorlagen und Startvorlagenversionen finden Sie unter [Steuern des Zugriffs auf Startvorlagen mit IAM-Berechtigungen](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zu Berechtigungen auf Ressourcenebene, mit denen Sie den Zugriff auf den `RunInstances`-Aufruf steuern können, finden Sie unter [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EC2](#) in der Referenz der Serviceberechtigung.

IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden

Anwendungen, die in Amazon EC2-Instances ausgeführt werden, benötigen Anmeldeinformationen, um auf andere AWS-Services zugreifen zu können. Verwenden Sie eine IAM-Rolle, um diese Anmeldeinformationen auf sichere Weise bereitstellen zu können. Die Rolle stellt temporäre Berechtigungen bereit, die von der Anwendung verwendet werden können, wenn sie auf andere AWS

-Ressourcen zugreift. Die Berechtigungen der Rolle bestimmen, welche Aktionen die Anwendung durchführen darf.

Für Instances in einer Auto Scaling-Gruppe müssen Sie eine Startkonfiguration oder Vorlage erstellen und ein Instance-Profil wählen, das mit den Instances verknüpft werden soll. Ein Instance-Profil ist ein Container für eine IAM-Rolle, mit dem Amazon EC2 einer Instance die IAM-Rolle übergibt, wenn die Instance gestartet wird. Erstellen Sie zunächst eine IAM-Rolle, die über alle für den Zugriff auf die Ressourcen erforderlichen Berechtigungen verfügt. AWS Erstellen Sie anschließend das Instance-Profil und weisen Sie diesem die Rolle zu.

Note

Als bewährte Methode empfehlen wir dringend, die Rolle so zu erstellen, dass sie über die Mindestberechtigungen für andere Benutzer verfügt AWS-Services , die für Ihre Anwendung erforderlich sind.

Inhalt

- [Voraussetzungen](#)
- [Erstellen einer Startvorlage](#)
- [Weitere Informationen finden Sie auch unter](#)

Voraussetzungen

Erstellen Sie die IAM-Rolle, die Ihre unter Amazon EC2 ausgeführte Anwendung verwenden kann. Wählen Sie die entsprechenden Berechtigungen, so dass die Anwendung, der die Rolle im Anschluss übertragen wird, die benötigten API-Aufrufe durchführen kann.

Wenn Sie die IAM-Konsole anstelle der AWS CLI oder einer der beiden verwenden AWS SDKs, erstellt die Konsole automatisch ein Instanzprofil und weist diesem denselben Namen zu wie der Rolle, der es entspricht.

So erstellen Sie eine IAM-Rolle (Konsole)

1. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
2. Wählen Sie im Navigationsbereich auf der linken Seite Roles (Rollen).
3. Wählen Sie Rolle erstellen aus.

4. Wählen Sie für Select trusted entity (Vertrauenswürdige Entität auswählen) die Option AWS - Service.
5. Wählen Sie für Ihren Anwendungsfall die Option EC2 und dann Next (Weiter) aus.
6. Wenn möglich, wählen Sie die Richtlinie aus, die für die Berechtigungsrichtlinie verwendet werden soll, oder wählen Create policy (Richtlinie erstellen), um eine neue Registerkarte im Browser zu öffnen und eine vollständig neue Richtlinie zu erstellen. Weitere Informationen finden Sie unter [Erstellen von IAM-Richtlinien](#) im IAM-Benutzerhandbuch. Nachdem Sie die Richtlinie erstellt haben, schließen Sie die Registerkarte und kehren zur ursprünglichen Registerkarte zurück. Aktivieren Sie die Kontrollkästchen neben den Berechtigungsrichtlinien, die der Service haben soll.
7. (Optional) Legen Sie eine Berechtigungsgrenze fest. Dies ist eine erweiterte Funktion, die für Servicerollen verfügbar ist. Weitere Informationen finden Sie unter [Berechtigungsgrenzen für IAM-Entitäten](#) im IAM-Benutzerhandbuch.
8. Wählen Sie Weiter aus.
9. Geben Sie auf der Seite Name, review, and create (Benennen, überprüfen und erstellen), für Role name (Rollenname) einen Rollennamen ein, mit dem der Zweck dieser Rolle einfach zu erkennen ist. Dieser Name muss innerhalb Ihres AWS-Konto eindeutig sein. Da andere AWS Ressourcen möglicherweise auf die Rolle verweisen, können Sie den Namen der Rolle nicht bearbeiten, nachdem sie erstellt wurde.
10. Prüfen Sie die Rolle und klicken Sie dann auf Create Role (Rolle erstellen).

IAM-Berechtigungen

Verwenden Sie eine identitätsbasierte IAM-Richtlinie, um den Zugriff auf Ihre neue IAM-Rolle zu steuern. Die `iam:PassRole`-Berechtigung ist für die IAM-Identität (Benutzer oder Rolle) erforderlich, die eine Auto-Scaling-Gruppe mithilfe einer Startvorlage erstellt oder aktualisiert, die ein Instance-Profil angibt.

Die folgende Beispielrichtlinie gewährt die Berechtigung, nur IAM-Rollen weiterzugeben, deren Name mit **qateam-** beginnt.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "iam:PassRole",  
      "Resource": "arn:aws:iam::*:*:role/qateam-*"    }  
  ]  
}
```

```

    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::111122223333:role/gateam-*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "ec2.amazonaws.com",
            "ec2.amazonaws.com.cn"
          ]
        }
      }
    }
  ]
}

```

Important

Informationen darüber, wie Amazon EC2 Auto Scaling Berechtigungen für die `iam:PassRole`-Aktion für eine Auto-Scaling-Gruppe validiert, die eine Startvorlage verwendet, finden Sie unter [Überprüfung der Berechtigungen für `ec2:RunInstances` und `iam:PassRole`](#).

Erstellen einer Startvorlage

Wenn Sie die Startvorlage mithilfe von erstellen AWS-Managementkonsole, wählen Sie im Abschnitt Erweiterte Details die Rolle aus dem IAM-Instanzprofil aus. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage mithilfe erweiterter Einstellungen](#).

Wenn Sie die Startvorlage mit dem [create-launch-template](#)Befehl von erstellen AWS CLI, geben Sie den Instanzprofilnamen Ihrer IAM-Rolle an, wie im folgenden Beispiel gezeigt.

```

aws ec2 create-launch-template --launch-template-name my-lt-with-instance-profile --
version-description version1 \
--launch-template-data
'{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro","IamInstanceProfile":
{"Name":"my-instance-profile"} }'

```

Weitere Informationen finden Sie auch unter

Weitere Informationen, die Sie beim Lernen von IAM-Rollen für Amazon EC2 unterstützen, finden Sie unter:

- [IAM-Rollen für Amazon EC2](#) im Amazon EC2 EC2-Benutzerhandbuch
- [Verwenden von Instance-Profilen](#) und [Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#) im IAM-Benutzerhandbuch

Compliance-Validierung für Amazon EC2 Auto Scaling

Informationen darüber, ob AWS-Service ein [AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt](#), finden Sie unter [Umfang nach Compliance-Programm AWS-Services unter](#) . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter [AWS Compliance-Programme AWS](#) .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Berichte herunterladen unter](#) .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der [AWS Sicherheitsdokumentation](#).

Compliance mit PCI DSS

Amazon EC2 Auto Scaling unterstützt die Verarbeitung, Speicherung und Übertragung von Kreditkartendaten durch einen Händler oder Dienstanbieter. Außerdem wurde seine Konformität mit dem Payment Card Industry (PCI) Data Security Standard (DSS) bestätigt. Weitere Informationen zu PCI DSS, einschließlich der Möglichkeit, eine Kopie des AWS PCI Compliance Package anzufordern, finden Sie unter [PCI DSS Level 1](#).

Informationen zum Erreichen der PCI-DSS-Konformität für Ihre AWS Workloads finden Sie im folgenden Compliance-Leitfaden:

- [Payment Card Industry Data Security Standard \(PCI DSS\) 3.2.1 auf AWS](#)

Amazon EC2 Auto Scaling und Schnittstellen-VPC-Endpunkte

Sie können die Sicherheit Ihrer VPC erhöhen, indem Sie Amazon EC2 Auto Scaling so konfigurieren, dass ein Schnittstellen-VPC-Endpunkt verwendet wird. Schnittstellenendpunkte werden von einer Technologie unterstützt AWS PrivateLink, die es Ihnen ermöglicht, privat auf Amazon EC2 Auto Scaling-APIs zuzugreifen, indem der gesamte Netzwerkverkehr zwischen Ihrer VPC und Amazon EC2 Auto Scaling auf das Netzwerk beschränkt wird. AWS Mit Schnittstellenendpunkten benötigen Sie außerdem kein Internet-Gateway, kein NAT-Gerät und kein Virtual Private Gateway.

Eine Konfiguration AWS PrivateLink ist nicht erforderlich, wird aber empfohlen. Weitere Informationen zu AWS PrivateLink VPC-Endpunkten finden Sie unter [Was ist? AWS PrivateLink](#) im Leitfaden.AWS PrivateLink

Themen

- [Erstellen eines Schnittstellen-VPC-Endpunkts](#)
- [Erstellen einer VPC-Endpunktrichtlinie](#)

Erstellen eines Schnittstellen-VPC-Endpunkts

Erstellen Sie einen Endpunkt für Amazon EC2 Auto Scaling mit dem folgenden Service-Namen:

```
com.amazonaws.region.autoscaling
```

Weitere Informationen finden Sie im AWS PrivateLink Handbuch unter [Zugreifen auf einen AWS Dienst über einen Schnittstellen-VPC-Endpunkt](#).

Sie müssen keine Amazon EC2 Auto Scaling-Einstellungen ändern. Amazon EC2 Auto Scaling ruft andere AWS Services entweder über Service-Endpunkte oder VPC-Endpunkte mit privater Schnittstelle auf, je nachdem, welche verwendet werden.

Erstellen einer VPC-Endpunktrichtlinie

Sie können Ihrem VPC-Endpunkt eine Richtlinie anfügen, um den Zugriff auf die Amazon EC2 Auto Scaling-API zu steuern. Die Richtlinie legt Folgendes fest:

- Prinzipal, der die Aktionen ausführen kann.
- Die Aktionen, die ausgeführt werden können.

- Die Ressource, auf der die Aktionen ausgeführt werden können.

Das folgende Beispiel zeigt eine VPC-Endpunktrichtlinie, die jedem Benutzer die Berechtigung zum Löschen einer Skalierungsrichtlinie über den Endpunkt verweigert. Die Beispielrichtlinie gewährt auch jedem die Berechtigung, alle anderen Aktionen auszuführen.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "autoscaling:DeleteScalingPolicy",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": "*"
    }
  ]
}
```

Weitere Informationen finden Sie unter [Kontrolle des Zugriffs auf VPC-Endpunkte mit Endpunktrichtlinien](#) im AWS PrivateLink Leitfaden.

Verwenden Sie diesen Service mit einem SDK AWS

AWS Software Development Kits (SDKs) sind für viele beliebte Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele
AWS CLI	AWS CLI Code-Beispiele
AWS SDK für Go	AWS SDK für Go Code-Beispiele
AWS SDK für Java	AWS SDK für Java Code-Beispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Code-Beispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Code-Beispiele
AWS SDK für .NET	AWS SDK für .NET Code-Beispiele
AWS SDK für PHP	AWS SDK für PHP Code-Beispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Code-Beispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Code-Beispiele
AWS SDK für Ruby	AWS SDK für Ruby Code-Beispiele
AWS SDK für Rust	AWS SDK für Rust Code-Beispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Code-Beispiele
AWS SDK für Swift	AWS SDK für Swift Code-Beispiele

Weitere Beispiele speziell für diesen Service finden Sie unter [Codebeispiele für Auto Scaling mit AWS SDKs](#).

Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link [Provide feedback \(Feedback geben\)](#) auswählen.

Codebeispiele für Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Auto Scaling mit einem AWS Software Development Kit (SDK) verwendet wird.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele

- [Grundlegende Beispiele für Auto Scaling mit AWS SDKs](#)
 - [Hallo, Auto Scaling](#)
 - [Lernen Sie die Grundlagen von Auto Scaling mit einem AWS SDK kennen](#)
 - [Aktionen für Auto Scaling mit AWS SDKs](#)
 - [Verwendung von AttachInstances mit einer CLI](#)
 - [Verwendung AttachLoadBalancerTargetGroups mit einem AWS SDK oder CLI](#)
 - [Verwendung von AttachLoadBalancers mit einer CLI](#)
 - [Verwendung von CompleteLifecycleAction mit einer CLI](#)
 - [Verwendung CreateAutoScalingGroup mit einem AWS SDK oder CLI](#)
 - [Verwendung von CreateLaunchConfiguration mit einer CLI](#)
 - [Verwendung von CreateOrUpdateTags mit einer CLI](#)
 - [Verwendung DeleteAutoScalingGroup mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteLaunchConfiguration mit einer CLI](#)
 - [Verwendung von DeleteLifecycleHook mit einer CLI](#)
 - [Verwendung von DeleteNotificationConfiguration mit einer CLI](#)

- [Verwendung von DeletePolicy mit einer CLI](#)
- [Verwendung von DeleteScheduledAction mit einer CLI](#)
- [Verwendung von DeleteTags mit einer CLI](#)
- [Verwendung von DescribeAccountLimits mit einer CLI](#)
- [Verwendung von DescribeAdjustmentTypes mit einer CLI](#)
- [Verwendung DescribeAutoScalingGroups mit einem AWS SDK oder CLI](#)
- [Verwendung DescribeAutoScalingInstances mit einem AWS SDK oder CLI](#)
- [Verwendung von DescribeAutoScalingNotificationTypes mit einer CLI](#)
- [Verwendung von DescribeLaunchConfigurations mit einer CLI](#)
- [Verwendung von DescribeLifecycleHookTypes mit einer CLI](#)
- [Verwendung von DescribeLifecycleHooks mit einer CLI](#)
- [Verwendung von DescribeLoadBalancers mit einer CLI](#)
- [Verwendung von DescribeMetricCollectionTypes mit einer CLI](#)
- [Verwendung von DescribeNotificationConfigurations mit einer CLI](#)
- [Verwendung von DescribePolicies mit einer CLI](#)
- [Verwendung DescribeScalingActivities mit einem AWS SDK oder CLI](#)
- [Verwendung von DescribeScalingProcessTypes mit einer CLI](#)
- [Verwendung von DescribeScheduledActions mit einer CLI](#)
- [Verwendung von DescribeTags mit einer CLI](#)
- [Verwendung von DescribeTerminationPolicyTypes mit einer CLI](#)
- [Verwendung von DetachInstances mit einer CLI](#)
- [Verwendung von DetachLoadBalancers mit einer CLI](#)
- [Verwendung DisableMetricsCollection mit einem AWS SDK oder CLI](#)
- [Verwendung EnableMetricsCollection mit einem AWS SDK oder CLI](#)
- [Verwendung von EnterStandby mit einer CLI](#)
- [Verwendung von ExecutePolicy mit einer CLI](#)
- [Verwendung von ExitStandby mit einer CLI](#)
- [Verwendung von PutLifecycleHook mit einer CLI](#)
- [Verwendung von PutNotificationConfiguration mit einer CLI](#)
- [Verwendung von PutScalingPolicy mit einer CLI](#)

- [Verwendung von PutScheduledUpdateGroupAction mit einer CLI](#)
- [Verwendung von RecordLifecycleActionHeartbeat mit einer CLI](#)
- [Verwendung von ResumeProcesses mit einer CLI](#)
- [Verwendung SetDesiredCapacity mit einem AWS SDK oder CLI](#)
- [Verwendung von SetInstanceHealth mit einer CLI](#)
- [Verwendung von SetInstanceProtection mit einer CLI](#)
- [Verwendung von SuspendProcesses mit einer CLI](#)
- [Verwendung TerminateInstancesInAutoScalingGroup mit einem AWS SDK oder CLI](#)
- [Verwendung UpdateAutoScalingGroup mit einem AWS SDK oder CLI](#)
- [Szenarien für Auto Scaling mit AWS SDKs](#)
 - [Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK](#)

Grundlegende Beispiele für Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon EC2 Auto Scaling mit AWS SDKs verwenden können.

Beispiele

- [Hallo, Auto Scaling](#)
- [Lernen Sie die Grundlagen von Auto Scaling mit einem AWS SDK kennen](#)
- [Aktionen für Auto Scaling mit AWS SDKs](#)
 - [Verwendung von AttachInstances mit einer CLI](#)
 - [Verwendung AttachLoadBalancerTargetGroups mit einem AWS SDK oder CLI](#)
 - [Verwendung von AttachLoadBalancers mit einer CLI](#)
 - [Verwendung von CompleteLifecycleAction mit einer CLI](#)
 - [Verwendung CreateAutoScalingGroup mit einem AWS SDK oder CLI](#)
 - [Verwendung von CreateLaunchConfiguration mit einer CLI](#)
 - [Verwendung von CreateOrUpdateTags mit einer CLI](#)
 - [Verwendung DeleteAutoScalingGroup mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteLaunchConfiguration mit einer CLI](#)
 - [Verwendung von DeleteLifecycleHook mit einer CLI](#)

- [Verwendung von DeleteNotificationConfiguration mit einer CLI](#)
- [Verwendung von DeletePolicy mit einer CLI](#)
- [Verwendung von DeleteScheduledAction mit einer CLI](#)
- [Verwendung von DeleteTags mit einer CLI](#)
- [Verwendung von DescribeAccountLimits mit einer CLI](#)
- [Verwendung von DescribeAdjustmentTypes mit einer CLI](#)
- [Verwendung DescribeAutoScalingGroups mit einem AWS SDK oder CLI](#)
- [Verwendung DescribeAutoScalingInstances mit einem AWS SDK oder CLI](#)
- [Verwendung von DescribeAutoScalingNotificationTypes mit einer CLI](#)
- [Verwendung von DescribeLaunchConfigurations mit einer CLI](#)
- [Verwendung von DescribeLifecycleHookTypes mit einer CLI](#)
- [Verwendung von DescribeLifecycleHooks mit einer CLI](#)
- [Verwendung von DescribeLoadBalancers mit einer CLI](#)
- [Verwendung von DescribeMetricCollectionTypes mit einer CLI](#)
- [Verwendung von DescribeNotificationConfigurations mit einer CLI](#)
- [Verwendung von DescribePolicies mit einer CLI](#)
- [Verwendung DescribeScalingActivities mit einem AWS SDK oder CLI](#)
- [Verwendung von DescribeScalingProcessTypes mit einer CLI](#)
- [Verwendung von DescribeScheduledActions mit einer CLI](#)
- [Verwendung von DescribeTags mit einer CLI](#)
- [Verwendung von DescribeTerminationPolicyTypes mit einer CLI](#)
- [Verwendung von DetachInstances mit einer CLI](#)
- [Verwendung von DetachLoadBalancers mit einer CLI](#)
- [Verwendung DisableMetricsCollection mit einem AWS SDK oder CLI](#)
- [Verwendung EnableMetricsCollection mit einem AWS SDK oder CLI](#)
- [Verwendung von EnterStandby mit einer CLI](#)
- [Verwendung von ExecutePolicy mit einer CLI](#)
- [Verwendung von ExitStandby mit einer CLI](#)
- [Verwendung von PutLifecycleHook mit einer CLI](#)
- [Verwendung von PutNotificationConfiguration mit einer CLI](#)

- [Verwendung von PutScalingPolicy mit einer CLI](#)
- [Verwendung von PutScheduledUpdateGroupAction mit einer CLI](#)
- [Verwendung von RecordLifecycleActionHeartbeat mit einer CLI](#)
- [Verwendung von ResumeProcesses mit einer CLI](#)
- [Verwendung SetDesiredCapacity mit einem AWS SDK oder CLI](#)
- [Verwendung von SetInstanceHealth mit einer CLI](#)
- [Verwendung von SetInstanceProtection mit einer CLI](#)
- [Verwendung von SuspendProcesses mit einer CLI](#)
- [Verwendung TerminateInstancesInAutoScalingGroup mit einem AWS SDK oder CLI](#)
- [Verwendung UpdateAutoScalingGroup mit einem AWS SDK oder CLI](#)

Hallo, Auto Scaling

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Auto Scaling beginnen.

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
}
```

```
static async Task Main(string[] args)
{
    var client = new AmazonAutoScalingClient();

    Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
    Console.WriteLine("Let's get a description of your Auto Scaling
groups.");

    var response = await client.DescribeAutoScalingGroupsAsync();

    if (response.AutoScalingGroups == null ||
response.AutoScalingGroups.Count == 0)
    {
        Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
groups.");
        return;
    }
    response.AutoScalingGroups.ForEach(autoScalingGroup =>
    {
        Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.AvailabilityZone}");
    }
}
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
  libraries for the AWS SDK.
  string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
    "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
  list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory
  for running and debugging.

  # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
  may need to uncomment this
  # and set the proper subdirectory to the
  executables' location.

  AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
  hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
```

```
#{AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_autoscaling.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 * A "Hello Autoscaling" starter application which initializes an Amazon EC2
 * Auto Scaling client and describes the
 * Amazon EC2 Auto Scaling groups.
 *
 * main function
 *
 * Usage: 'hello_autoscaling'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
        }
```

```

        Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
            autoscalingClient.DescribeAutoScalingGroups(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
&autoScalingGroups =
                outcome.GetResult().GetAutoScalingGroups();
            for (auto &group: autoScalingGroups) {
                groupNames.push_back(group.GetAutoScalingGroupName());
            }
            nextToken = outcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
"
                << outcome.GetError().GetMessage()
                << std::endl;
            result = 1;
            break;
        }
    } while (!nextToken.empty());

    std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
std::endl;
    for (auto &groupName: groupNames) {
        std::cout << "AutoScaling group: " << groupName << std::endl;
    }
}

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}

```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) unter AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
            autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
        });
    }
}
```

```
    });  
  }  
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function helloService()  
{  
    $autoScalingClient = new AutoScalingClient([  
        'region' => 'us-west-2',  
        'version' => 'latest',  
        'profile' => 'default',  
    ]);  
  
    $groups = $autoScalingClient->describeAutoScalingGroups([]);  
    var_dump($groups);  
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import boto3

def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
    client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
        groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
            {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")

if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
# operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:                #{group.auto_scaling_group_arn}")
      end
    end
  end
end
```

```
        @logger.info("  Min/max/desired:      #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
  let resp = client.describe_auto_scaling_groups().send().await?;

  println!("Groups:");

  let groups = resp.auto_scaling_groups();

  for group in groups {
    println!(
      "Name: {}",
      group.auto_scaling_group_name().unwrap_or("Unknown")
    );
  }
}
```

```
println!(
    "Arn:  {}",
    group.auto_scaling_group_arn().unwrap_or("unknown"),
);
println!("Zones: {:?}", group.availability_zones(),);
println!();
}

println!("Found {} group(s)", groups.len());

Ok(())
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Lernen Sie die Grundlagen von Auto Scaling mit einem AWS SDK kennen

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen einer Amazon-EC2-Auto-Scaling-Gruppe mit einer Startvorlage und Availability Zones sowie Abrufen von Informationen zu laufenden Instances
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann Ressourcen.

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;

using Amazon.EC2;
using Microsoft.Extensions.Configuration;
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
                        LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
                        LogLevel.Trace))
            .Build();
    }
}
```

```
.ConfigureServices( (_, services) =>
services.AddAWSService<IAmazonAutoScaling>()
    .AddAWSService<IAmazonCloudWatch>()
    .AddAWSService<IAmazonEC2>()
    .AddTransient<AutoScalingWrapper>()
    .AddTransient<CloudWatchWrapper>()
    .AddTransient<EC2Wrapper>()
    .AddTransient<UIWrapper>()
)
.Build();

var autoScalingWrapper =
host.Services.GetRequiredService<AutoScalingWrapper>();
var cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();
var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

var configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from .json file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

var imageId = configuration["ImageId"];
var instanceType = configuration["InstanceType"];
var launchTemplateName = configuration["LaunchTemplateName"];

launchTemplateName += Guid.NewGuid().ToString();

// The name of the Auto Scaling group.
var groupName = configuration["GroupName"];

uiWrapper.DisplayTitle("Auto Scaling Basics");
uiWrapper.DisplayAutoScalingBasicsDescription();

// Create the launch template and save the template Id to use when
deleting the
// launch template at the end of the application.
var launchTemplateId = await
ec2Wrapper.CreateLaunchTemplateAsync(imageId!, instanceType!,
launchTemplateName);
```

```
it.
    // Confirm that the template was created by asking for a description of
    await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

    uiWrapper.PressEnter();

    var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

    Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
    await autoScalingWrapper.CreateAutoScalingGroupAsync(
        groupName!,
        launchTemplateName,
        availabilityZones[0].ZoneName);

    // Keep checking the details of the new group until its lifecycle state
    // is "InService".
    Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

    List<AutoScalingInstanceDetails> instanceDetails;

    do
    {
        instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
    }
    while (instanceDetails.Count <= 0);

    Console.WriteLine($"Auto scaling group {groupName} successfully
created.");
    Console.WriteLine($"{instanceDetails.Count} instances were created for
the group.");

    // Display the details of the Auto Scaling group.
    instanceDetails.ForEach(detail =>
    {
        Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
    });

    uiWrapper.PressEnter();

    uiWrapper.DisplayTitle("Metrics collection");
    Console.WriteLine($"Enable metrics collection for {groupName}");
    await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);
```

```
// Show the metrics that are collected for the group.

// Update the maximum size of the group to three instances.
Console.WriteLine("--- Update the Auto Scaling group to increase max size
to 3 ---");
int maxSize = 3;
await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

Console.WriteLine("--- Describe all Auto Scaling groups to show the
current state of the group ---");
var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

uiWrapper.DisplayGroupDetails(groups!);

uiWrapper.PressEnter();

uiWrapper.DisplayTitle("Describe account limits");
await autoScalingWrapper.DescribeAccountLimitsAsync();

uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");

uiWrapper.DisplayTitle("Set desired capacity");
int desiredCapacity = 2;
await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

Console.WriteLine("Get the two instance Id values");

// Empty the group before getting the details again.
groups.Clear();
groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
if (groups.Any())
{
    foreach (AutoScalingGroup group in groups)
    {
        Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
        Console.WriteLine($"The group ARN is
{group.AutoScalingGroupARN}");
        var instances = group.Instances;
```

```
        foreach (Amazon.AutoScaling.Model.Instance instance in instances)
        {
            Console.WriteLine($"The instance id is
{instance.InstanceId}");
            Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
        }
    }

    uiWrapper.DisplayTitle("Scaling Activities");
    Console.WriteLine("Let's list the scaling activities that have occurred
for the group.");
    var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
    if (activities.Any())
    {
        activities.ForEach(activity =>
        {
            Console.WriteLine($"The activity Id is {activity.ActivityId}");
            Console.WriteLine($"The activity details are
{activity.Details}");
        });
    }

    // Display the Amazon CloudWatch metrics that have been collected.
    var metrics = await
cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
    if (metrics.Any())
    {
        Console.WriteLine($"Metrics collected for {groupName}:");
        metrics.ForEach(metric =>
        {
            Console.WriteLine($"Metric name: {metric.MetricName}\t");
            Console.WriteLine($"Namespace: {metric.Namespace}");
        });
    }

    var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
    if (dataPoints.Any())
    {
        Console.WriteLine("Details for the metrics collected:");
        dataPoints.ForEach(detail => { Console.WriteLine(detail); });
    }
}
```

```
    }

    // Disable metrics collection.
    Console.WriteLine("Disabling the collection of metrics for
{groupName}.");
    var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

    if (success)
    {
        Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
    }
    else
    {
        Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
    }

    // Terminate all instances in the group.
    uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
    Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

    if (groups is not null)
    {
        groups.ForEach(group =>
        {
            // Only delete instances in the AutoScaling group we created.
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(async instance =>
                {
                    await
autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                });
            }
        });
    }

    // After all instances are terminated, delete the group.
    uiWrapper.DisplayTitle("Clean up resources");
    Console.WriteLine("Deleting the Auto Scaling group.");
    await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);
```

```
        // Delete the launch template.
        var deletedLaunchTemplateName = await
ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }

        Console.WriteLine("The demo is now concluded.");
    }
}

namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
    {
        Console.WriteLine("This code example performs the following
operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
        Console.WriteLine(" 7. Changes the desired capacity of the Auto
Scaling");
        Console.WriteLine("    group to use an additional instance.");
    }
}
```

```

        Console.WriteLine(" 8. Shows that there are now instances in the
group.");
        Console.WriteLine(" 9. Lists the scaling activities that have occurred
for the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that
have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>
        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }
}

```

```
/// <summary>
/// Pad a string with spaces to center it on the console display.
/// </summary>
/// <param name="strToCenter">The string to be centered.</param>
/// <returns>The padded string.</returns>
public string CenterString(string strToCenter)
{
    var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
    var leftPad = new string(' ', padAmount);
    return $"{leftPad}{strToCenter}";
}

/// <summary>
/// Display a line of hyphens, the centered text of the title and another
/// line of hyphens.
/// </summary>
/// <param name="strTitle">The string to be displayed.</param>
public void DisplayTitle(string strTitle)
{
    Console.WriteLine(SepBar);
    Console.WriteLine(CenterString(strTitle));
    Console.WriteLine(SepBar);
}

/// <summary>
/// Display a countdown and wait for a number of seconds.
/// </summary>
/// <param name="numSeconds">The number of seconds to wait.</param>
public void WaitABit(int numSeconds, string msg)
{
    Console.WriteLine(msg);

    // Wait for the requested number of seconds.
    for (int i = numSeconds; i > 0; i--)
    {
        System.Threading.Thread.Sleep(1000);
        Console.Write($"{i}...");
    }

    PressEnter();
}
}
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Startvorlagen und Metriken zu verwalten. Diese Funktionen umfassen Auto Scaling, Amazon EC2 und CloudWatch Aktionen.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)
    {
        _amazonAutoScaling = amazonAutoScaling;
    }

    /// <summary>
    /// Create a new Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name to use for the new Auto Scaling
    /// group.</param>
    /// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
    /// launch template to use to create instances in the group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        string availabilityZone)
```

```
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
    {
        availabilityZone,
    };

    var request = new CreateAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        AvailabilityZones = zoneList,
        LaunchTemplate = templateSpecification,
        MaxSize = 6,
        MinSize = 1
    };
    try
    {
        var response = await
        _amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AlreadyExistsException)
    {
        Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
        return true;
    }
}

/// <summary>
/// Retrieve information about Amazon EC2 Auto Scaling quotas to the
/// active AWS account.
/// </summary>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DescribeAccountLimitsAsync()
{
    var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
```

```
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
response.MaxNumberOfAutoScalingGroups);
        Console.WriteLine("The current number of Auto Scaling groups is " +
response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
_amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }

    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
```

```
public async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    string groupName)
{
    var groups = await DescribeAutoScalingGroupsAsync(groupName);
    var instanceIds = new List<string>();
    var instanceDetails = new List<AutoScalingInstanceDetails>();
    if (groups != null)
    {
        groups.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            });
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response =
            await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                scalingGroupsRequest);
        if (response.AutoScalingInstances != null)
        {
            instanceDetails = response.AutoScalingInstances;
        }
    }

    return instanceDetails;
}

/// <summary>
/// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
/// </summary>
```

```
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
    public async Task<List<AutoScalingGroup>> DescribeAutoScalingGroupsAsync(
        string groupName)
    {
        var groups = new List<AutoScalingGroup>();
        var groupList = new List<string>
            {
                groupName,
            };

        var request = new DescribeAutoScalingGroupsRequest
        {
            AutoScalingGroupNames = groupList,
        };

        var response = await
        _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
        if (response.AutoScalingGroups != null)
        {
            groups = response.AutoScalingGroups;
        }

        return groups;
    }

    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };
    }
```

```
        var response = await
_amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }

    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
        {
            "GroupMaxSize",
        };
    }
}
```

```
var collectionRequest = new EnableMetricsCollectionRequest
{
    AutoScalingGroupName = groupName,
    Metrics = listMetrics,
    Granularity = "1Minute",
};

var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Set the desired capacity of an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="desiredCapacity">The desired capacity for the Auto
/// Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> SetDesiredCapacityAsync(
    string groupName,
    int desiredCapacity)
{
    var capacityRequest = new SetDesiredCapacityRequest
    {
        AutoScalingGroupName = groupName,
        DesiredCapacity = desiredCapacity,
    };

    var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
    Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Terminate all instances in the Auto Scaling group in preparation for
/// deleting the group.
/// </summary>
```

```
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
        _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }

    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
```

```
};

var groupRequest = new UpdateAutoScalingGroupRequest
{
    MaxSize = maxSize,
    AutoScalingGroupName = groupName,
    LaunchTemplate = templateSpecification,
};

var response = await
_amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
    return true;
}
else
{
    return false;
}
}

namespace AutoScalingActions;

using Amazon.EC2;
using Amazon.EC2.Model;

public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }
}
```

```
/// <summary>
/// Create a new Amazon EC2 launch template.
/// </summary>
/// <param name="imageId">The image Id to use for instances launched
/// using the Amazon EC2 launch template.</param>
/// <param name="instanceType">The type of EC2 instances to create.</param>
/// <param name="launchTemplateName">The name of the launch template.</param>
/// <returns>Returns the TemplateID of the new launch template.</returns>
public async Task<string> CreateLaunchTemplateAsync(
    string imageId,
    string instanceType,
    string launchTemplateName)
{
    var request = new CreateLaunchTemplateRequest
    {
        LaunchTemplateData = new RequestLaunchTemplateData
        {
            ImageId = imageId,
            InstanceType = instanceType,
        },
        LaunchTemplateName = launchTemplateName,
    };

    var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

    return response.LaunchTemplate.LaunchTemplateId;
}

/// <summary>
/// Delete an Amazon EC2 launch template.
/// </summary>
/// <param name="launchTemplateId">The TemplateId of the launch template to
/// delete.</param>
/// <returns>The name of the EC2 launch template that was deleted.</returns>
public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
{
    var request = new DeleteLaunchTemplateRequest
    {
        LaunchTemplateId = launchTemplateId,
    };

    var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
    return response.LaunchTemplate.LaunchTemplateName;
}
```

```
    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string
launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.WriteLine($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
    }

    /// <summary>
    /// Retrieve the availability zones for the current region.
    /// </summary>
    /// <returns>A collection of availability zones.</returns>
    public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
    {
        var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());

        return response.AvailabilityZones;
    }
}
```

```
namespace AutoScalingActions;

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</
returns>
    public async Task<List<Metric>> GetCloudWatchMetricsAsync(string groupName)
    {
        var metrics = new List<Metric>();
        var filter = new DimensionFilter
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        };

        var request = new ListMetricsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = new List<DimensionFilter> { filter },
            Namespace = "AWS/AutoScaling",
        };
    }
}
```

```
};

var response = await _amazonCloudWatch.ListMetricsAsync(request);
if (response.Metrics != null)
{
    metrics = response.Metrics;
}
return metrics;
}

/// <summary>
/// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A list of data points.</returns>
public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
{
    var dataPoints = new List<Datapoint>();
    var metricDimensions = new List<Dimension>
    {
        new Dimension
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        },
    };

    // The start time will be yesterday.
    var startTime = DateTime.UtcNow.AddDays(-1);

    var request = new GetMetricStatisticsRequest
    {
        MetricName = "AutoScalingGroupName",
        Dimensions = metricDimensions,
        Namespace = "AWS/AutoScaling",
        Period = 60, // 60 seconds.
        Statistics = new List<string>() { "Minimum" },
        StartTimeUtc = startTime,
        EndTimeUtc = DateTime.UtcNow,
    };

    var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);
    if (response.Datapoints != null)
```

```
    {  
        dataPoints = response.Datapoints;  
    }  
  
    return dataPoints;  
}  
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

C++

SDK für C++

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
//! Routine which demonstrates using an Auto Scaling group  
//! to manage Amazon EC2 instances.  
/*!
```

```

    \sa groupsAndInstancesScenario()
    \param clientConfig: AWS client configuration.
    \return bool: Successful completion.
*/
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;
    std::cout
        << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
Scaling "
        << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
        << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
        "Would you like to use an existing EC2 launch template (y/n)? ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
            "Enter the name of the existing EC2 launch template. ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateName(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
            ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                << "' exists by calling DescribeLaunchTemplate." <<
std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template.
"
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }
    else { // 2. Or create a new EC2 launch template.

```

```
templateName = askQuestion("Enter the name for a new EC2 launch template:");

Aws::EC2::Model::CreateLaunchTemplateRequest request;
request.SetLaunchTemplateName(templateName);

Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;
requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

request.SetLaunchTemplateData(requestLaunchTemplateData);

Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
    ec2Client.CreateLaunchTemplate(request);

if (outcome.IsSuccess()) {
    std::cout << "The EC2 launch template '" << templateName << "' was
created."
                << std::endl;
}
else if (outcome.GetError().GetExceptionName() ==
        "InvalidLaunchTemplateName.AlreadyExistsException") {
    std::cout << "The EC2 template '" << templateName << "' already
exists"
                << std::endl;
}
else {
    std::cerr << "Error with EC2::CreateLaunchTemplate. "
                << outcome.GetError().GetMessage()
                << std::endl;
}
}

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
std::cout << "Let's create an Auto Scaling group." << std::endl;
Aws::String groupName = askQuestion(
    "Enter a name for the Auto Scaling group: ");
// 3. Retrieve a list of EC2 Availability Zones.
Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
{
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
        ec2Client.DescribeAvailabilityZones(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout
            << "EC2 instances can be created in the following
Availability Zones:"
            << std::endl;

        availabilityZones = outcome.GetResult().GetAvailabilityZones();
        for (size_t i = 0; i < availabilityZones.size(); ++i) {
            std::cout << "    " << i + 1 << ". "
                << availabilityZones[i].GetZoneName() << std::endl;
        }
    }
    else {
        std::cerr << "Error with EC2::DescribeAvailabilityZones. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources("", templateName, autoScalingClient, ec2Client);
        return false;
    }
}

int availabilityZoneChoice = askQuestionForIntRange(
    "Choose an Availability Zone: ", 1,
    static_cast<int>(availabilityZones.size()));
// 4. Create an Auto Scaling group with the specified Availability Zone.
{
    Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
    Aws::Vector<Aws::String> availabilityGroupZones;
    availabilityGroupZones.push_back(
        availabilityZones[availabilityZoneChoice - 1].GetZoneName());
    request.SetAvailabilityZones(availabilityGroupZones);
    request.SetMaxSize(1);
    request.SetMinSize(1);

    Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
    launchTemplateSpecification.SetLaunchTemplateName(templateName);
    request.SetLaunchTemplate(launchTemplateSpecification);

    Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
        autoScalingClient.CreateAutoScalingGroup(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Created Auto Scaling group '" << groupName << "'..."
            << std::endl;
    }
    else if (outcome.GetError().GetErrorType() ==
        Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
        std::cout << "Auto Scaling group '" << groupName << "' already
exists."
            << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources("", templateName, autoScalingClient, ec2Client);
        return false;
    }
}

Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
    autoScalingClient)) {
    std::cout << "Here is the Auto Scaling group description." << std::endl;
    if (!autoScalingGroups.empty()) {
        logAutoScalingGroupInfo(autoScalingGroups);
    }
}
else {
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

std::cout
    << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
    << std::endl;
if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

bool enableMetrics = askYesNoQuestion(
    "Do you want to collect metrics about the A"
    "Auto Scaling group during this demo (y/n)? ");
```

```
// 7. Optionally enable metrics collection for the Auto Scaling group.
if (enableMetrics) {
    Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    request.AddMetrics("GroupMinSize");
    request.AddMetrics("GroupMaxSize");
    request.AddMetrics("GroupDesiredCapacity");
    request.AddMetrics("GroupInServiceInstances");
    request.AddMetrics("GroupTotalInstances");
    request.SetGranularity("1Minute");

    Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
        autoScalingClient.EnableMetricsCollection(request);
    if (outcome.IsSuccess()) {
        std::cout << "Auto Scaling metrics have been enabled."
                  << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
    "' from 1 to 3." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 8. Update the Auto Scaling group, setting a new maximum size.
{
    Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetMaxSize(3);

    Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
        autoScalingClient.UpdateAutoScalingGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                  << outcome.GetError().GetMessage()

```

```

        << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        const auto &instances = autoScalingGroups[0].GetInstances();
        std::cout
            << "The group still has one running EC2 instance, but it can
have up to 3.\n"
            << std::endl;
        logAutoScalingGroupInfo(autoScalingGroups);
    }
    else {
        std::cerr
            << "No EC2 launch groups were retrieved from DescribeGroup
request."
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
    << std::endl;
std::cout << "Let's update the desired capacity in '" << groupName <<
    "' from 1 to 2." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 9. Update the Auto Scaling group, setting a new desired capacity.
{
    Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetDesiredCapacity(2);

    Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
        autoScalingClient.SetDesiredCapacity(request);

    if (!outcome.IsSuccess()) {

```

```
        std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                << outcome.GetError().GetMessage()
                << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        std::cout
            << "Here is the current state of the group." << std::endl;
        logAutoScalingGroupInfo(autoScalingGroups);
    }
    else {
        std::cerr
            << "No EC2 launch groups were retrieved from DescribeGroup
request."
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
waitForInstances(groupName, autoScalingGroups, autoScalingClient);

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
        << std::endl;

std::cout << "Let's terminate one of the EC2 instances in " << groupName <<
"."
        << std::endl;
std::cout << "Because the desired capacity is 2, another EC2 instance will
start "
        << "to replace the terminated EC2 instance."
        << std::endl;
std::cout << "The currently running EC2 instances are:" << std::endl;

if (autoScalingGroups.empty()) {
```

```
        std::cerr << "Error describing groups. No groups returned." << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    int instanceNumber = 1;
    Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
        autoScalingGroups[0].GetInstances());
    for (const Aws::String &instanceID: instanceIDs) {
        std::cout << "    " << instanceNumber << ". " << instanceID << std::endl;
        ++instanceNumber;
    }

    instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to
stop? ",
                                           1,
                                           static_cast<int>(instanceIDs.size()));

    // 10. Terminate an EC2 instance in the Auto Scaling group.
    {
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
            autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                << instanceIDs[instanceNumber - 1] << "' to terminate..."
                << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }
```

```

    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
        << std::endl;
    std::cout << "Let's get a report of scaling activities for EC2 launch group
""
        << groupName << "."
        << std::endl;
    askQuestion("Press enter to continue: ", alwaysTrueTest);
    // 11. Get a description of activities for the Auto Scaling group.
    {
        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                    outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
                    << outcome.GetError().GetMessage()
                    << std::endl;
                cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
                return false;
            }
        } while (!nextToken.empty());
    }

```

```
std::cout << "Found " << allActivities.size() << " activities."
    << std::endl;
std::cout << "Activities are ordered with the most recent first."
    << std::endl;
for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
    std::cout << activity.GetDescription() << std::endl;
    std::cout << activity.GetDetails() << std::endl;
}
}

if (enableMetrics) {
    if (!logAutoScalingMetrics(groupName, clientConfig)) {
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

std::cout << "Let's clean up." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);

// 13. Disable metrics collection if enabled.
if (enableMetrics) {
    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
        autoScalingClient.DisableMetricsCollection(request);

    if (outcome.IsSuccess()) {
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}
```

```

    return cleanupResources(groupName, templateName, autoScalingClient,
        ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
        const
    Aws::AutoScaling::AutoScalingClient &client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {

```

```
        break;
    }
    else {
        if ((count % 5) == 0) {
            std::cout << "No instance IDs returned for group." <<
std::endl;
        }

        continue;
    }
}

// 6. Check lifecycle state of the instances using
DescribeAutoScalingInstances.
Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
request.SetInstanceIds(instanceIDs);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
    client.DescribeAutoScalingInstances(request);

if (outcome.IsSuccess()) {
    const
Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
        outcome.GetResult().GetAutoScalingInstances();
    ready = instancesDetails.size() >= desiredCapacity;
    for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails
&details: instancesDetails) {
        if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
            ready = false;
            break;
        }
    }
    // Log the status while waiting.
    if (((count % 5) == 1) || ready) {
        logInstancesLifecycleState(instancesDetails);
    }
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
```

```

    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const
                                           Aws::AutoScaling::AutoScalingClient &autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client)
{
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
        (askYesNoQuestion(
            Aws::String("Delete the Auto Scaling group '" + groupName +
                "' (y/n)?")))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout
                    << "The minimum size and desired capacity of the Auto
Scaling group "

```

```

        << "was set to zero before terminating the instances."
        << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
            << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
}

Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
    autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
            autoScalingGroups[0].GetInstances());
        for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
        request.SetInstanceId(instanceID);
        request.SetShouldDecrementDesiredCapacity(true);

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =
autoScalingClient.TerminateInstanceInAutoScalingGroup(
            request);

            if (outcome.IsSuccess()) {
                std::cout << "Initiating termination of EC2 instance '"
                    << instanceID << "'." << std::endl;
            }
            else {
                std::cerr
                    << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                    << outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
        }
    }

    std::cout

```

```

        << "Waiting for the EC2 instances to terminate before
deleting the "
        << "Auto Scaling group..." << std::endl;
        waitForInstances(groupName, autoScalingGroups, autoScalingClient);
    }

    {
        Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
            autoScalingClient.DeleteAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }
}

// 15. Delete the EC2 launch template.
if (!templateName.empty() && (askYesNoQuestion(
    Aws::String("Delete the EC2 launch template '" + templateName +
        "' (y/n)?"))) {
    Aws::EC2::Model::DeleteLaunchTemplateRequest request;
    request.SetLaunchTemplateName(templateName);

    Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
        ec2Client.DeleteLaunchTemplate(request);

    if (outcome.IsSuccess()) {
        std::cout << "EC2 launch template '" << templateName << "' was
deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with EC2::DeleteLaunchTemplate. "

```

```
        << outcome.GetError().GetMessage()
        << std::endl;
    result = false;
    }
}

return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
    const Aws::AutoScaling::AutoScalingClient
&client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
 * templates.html#create-launch-template
```

```
*
* This code example performs the following operations:
* 1. Creates an Auto Scaling group using an AutoScalingWaiter.
* 2. Gets a specific Auto Scaling group and returns an instance Id value.
* 3. Describes Auto Scaling with the Id value.
* 4. Enables metrics collection.
* 5. Update an Auto Scaling group.
* 6. Describes Account details.
* 7. Describe account details"
* 8. Updates an Auto Scaling group to use an additional instance.
* 9. Gets the specific Auto Scaling group and gets the number of instances.
* 10. List the scaling activities that have occurred for the group.
* 11. Terminates an instance in the Auto Scaling group.
* 12. Stops the metrics collection.
* 13. Deletes the Auto Scaling group.
*/

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    private static final String ROLES_STACK = "MyCdkAutoScaleStack";
    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <groupName>

            Where:
                groupName - The name of the Auto Scaling group.
            """;

        String groupName = "MyAutoScalingGroup2";
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_WEST_2)
            .build();

        Ec2Client ec2 = Ec2Client.builder()
            .region(Region.US_WEST_2)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
        System.out.println(DASHES);
    }
}
```

```
System.out.println(DASHES);
System.out.println("First, we will create a launch template using a
CloudFormation script");
CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
String launchTemplateName = stackOutputs.get("LaunchTemplateNameOutput");
String vpcZoneId = getVPC(ec2);
updateTemplate(ec2, launchTemplateName );
System.out.println("The VPC zone id created by the CloudFormation stack
is"+vpcZoneId);

System.out.println("1. Create an Auto Scaling group named " + groupName);
createAutoScalingGroup(autoScalingClient, ec2, groupName,
launchTemplateName, vpcZoneId);

System.out.println(
    "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
Thread.sleep(60000);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get Auto Scale group Id value");
String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
if (instanceId.compareTo("") == 0) {
    System.out.println("Error - no instance Id value");
    System.exit(1);
} else {
    System.out.println("The instance Id value is " + instanceId);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
describeAutoScalingInstance(autoScalingClient, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enable metrics collection " + instanceId);
enableMetricsCollection(autoScalingClient, groupName);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Update an Auto Scaling group to update max size to
3");
updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Describe Auto Scaling groups");
describeAutoScalingGroups(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Describe account details");
describeAccountLimits(autoScalingClient);
System.out.println(
    "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
Thread.sleep(60000);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Set desired capacity to 2");
setDesiredCapacity(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get the two instance Id values and state");
getSpecificAutoScalingGroups(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. List the scaling activities that have occurred
for the group");
describeScalingActivities(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Terminate an instance in the Auto Scaling
group");
terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("12. Stop the metrics collection");
disableMetricsCollection(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Delete the Auto Scaling group and cloud formation
resources");
CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
deleteAutoScalingGroup(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed.");
System.out.println(DASHES);

autoScalingClient.close();
}

public static String getVPC(Ec2Client ec2) {
    try {
        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
            .filters(f -> f.name("isDefault").values("true"))
            .build();

        DescribeVpcsResponse response = ec2.describeVpcs(request);

        if (!response.vpcs().isEmpty()) {
            Vpc defaultVpc = response.vpcs().get(0);
            System.out.println("Default VPC ID: " + defaultVpc.vpcId());
            return defaultVpc.vpcId();
        } else {
            System.out.println("No default VPC found.");
            return null; // Return null if no default VPC is found
        }

    } catch (Ec2Exception e) {
        System.err.println("EC2 error: " +
e.awsErrorDetails().errorMessage());
        return null; // Return null in case of an error
    }
}
```

```
public static void updateTemplate(Ec2Client ec2, String launchTemplateName ) {
    // Step 1: Create new launch template version
    String newAmiId = "ami-0025f0db847eb6254";
    RequestLaunchTemplateData launchTemplateData =
RequestLaunchTemplateData.builder()
        .imageId(newAmiId)
        .build();

    CreateLaunchTemplateVersionRequest createVersionRequest =
CreateLaunchTemplateVersionRequest.builder()
        .launchTemplateName(launchTemplateName)
        .versionDescription("Updated with valid AMI")
        .sourceVersion("1")
        .launchTemplateData(launchTemplateData)
        .build();

    CreateLaunchTemplateVersionResponse createResponse =
ec2.createLaunchTemplateVersion(createVersionRequest);
    int newVersionNumber =
createResponse.launchTemplateVersion().versionNumber().intValue();

    // Step 2: Modify default version
    ModifyLaunchTemplateRequest modifyRequest =
ModifyLaunchTemplateRequest.builder()
        .launchTemplateName(launchTemplateName)
        .defaultVersion(String.valueOf(newVersionNumber))
        .build();

    ec2.modifyLaunchTemplate(modifyRequest);
    System.out.println("Updated launch template to version " +
newVersionNumber + " with AMI " + newAmiId);
}

public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response = autoScalingClient
```

```
        .describeScalingActivities(ScalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity : activities) {
            System.out.println("The activity Id is " +
activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
                                           Ec2Client ec2Client,
                                           String groupName,
                                           String launchTemplateName,
                                           String vpcId) {
    try {
        // Step 1: Get one subnet ID in the given VPC
        DescribeSubnetsRequest subnetRequest =
DescribeSubnetsRequest.builder()
```

```
        .filters(Filter.builder().name("vpc-
id").values(vpcId).build())
        .build();

    DescribeSubnetsResponse subnetResponse =
ec2Client.describeSubnets(subnetRequest);

    if (subnetResponse.subnets().isEmpty()) {
        throw new RuntimeException("No subnets found in VPC: " + vpcId);
    }

    String subnetId = subnetResponse.subnets().get(0).subnetId(); // Use
first subnet
    System.out.println("Using subnet: " + subnetId);

    // Step 2: Create launch template reference
    LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
        .launchTemplateName(launchTemplateName)
        .build();

    // Step 3: Create Auto Scaling group
    CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .launchTemplate(templateSpecification)
        .minSize(1)
        .maxSize(1)
        .vpcZoneIdentifier(subnetId) // Correct: subnet ID, not VPC
ID
        .build();

    autoScalingClient.createAutoScalingGroup(request);

    // Step 4: Wait until group is created
    AutoScalingWaiter waiter = autoScalingClient.waiter();
    DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
        waiter.waitUntilGroupExists(groupsRequest);
```

```
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (Ec2Exception | AutoScalingException e) {
        System.err.println("Error: " + e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .maxRecords(10)
        .build();
```

```
        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("*** The service to use for the health checks:
" + group.healthCheckType());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response = autoScalingClient
            .describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " +
group.autoScalingGroupARN());
            List<Instance> instances = group.instances();

            for (Instance instance : instances) {
                instanceId = instance.instanceId();
                System.out.println("The instance id is " + instanceId);
                System.out.println("The lifecycle state is " +
instance.lifecycleState());
            }
        }

        return instanceId;
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}

public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void describeAccountLimits(AutoScalingClient autoScalingClient)
{
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is " +
response.numberOfWorkingAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
            .waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
    }
}
```

```
        System.out.println("You successfully updated the auto scaling group
" + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel](#) einrichten und ausführen.

```
suspend fun main(args: Array<String>) {  
    val usage = ""  
    Usage:  
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>  
  
    Where:  
        groupName - The name of the Auto Scaling group.  
        launchTemplateName - The name of the launch template.  
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-  
        linked role that the Auto Scaling group uses.
```

```
    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances
in the Auto Scaling group can be created.
    """"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("***** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("***** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("***** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("***** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("***** Describe all Auto Scaling groups to show the current state of
the groups")
    describeAutoScalingGroups(groupName)

    println("***** Describe account details")
```

```
describeAccountLimits()

println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
delay(60000)

println("**** Set desired capacity to 2")
setDesiredCapacity(groupName)

println("**** Get the two instance Id values and state")
getAutoScalingGroups(groupName)

println("**** List the scaling activities that have occurred for the group")
describeScalingActivities(groupName)

println("**** Terminate an instance in the Auto Scaling group")
terminateInstanceInAutoScalingGroup(instanceId)

println("**** Stop the metrics collection")
disableMetricsCollection(groupName)

println("**** Delete the Auto Scaling group")
deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
```

```
        autoScalingGroupName = groupName
        metrics = listOf("GroupMaxSize")
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
            response.activities?.forEach { activity ->
                println("The activity Id is ${activity.activityId}")
                println("The activity details are ${activity.details}")
            }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
            response.autoScalingGroups?.forEach { group ->
                println("The group name is ${group.autoScalingGroupName}")
                println("The group ARN is ${group.autoScalingGroupArn}")
                group.instances?.forEach { instance ->
                    println("The instance id is ${instance.instanceId}")
                    println("The lifecycle state is " + instance.lifecycleState)
                }
            }
    }
}
```

```
    }  
  }  
}  
  
suspend fun setDesiredCapacity(groupName: String) {  
  val capacityRequest =  
    SetDesiredCapacityRequest {  
      autoScalingGroupName = groupName  
      desiredCapacity = 2  
    }  
  
  AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
    autoScalingClient.setDesiredCapacity(capacityRequest)  
    println("You set the DesiredCapacity to 2")  
  }  
}  
  
suspend fun updateAutoScalingGroup(  
  groupName: String,  
  launchTemplateNameVal: String,  
  serviceLinkedRoleARNVal: String,  
) {  
  val templateSpecification =  
    LaunchTemplateSpecification {  
      launchTemplateName = launchTemplateNameVal  
    }  
  
  val groupRequest =  
    UpdateAutoScalingGroupRequest {  
      maxSize = 3  
      serviceLinkedRoleArn = serviceLinkedRoleARNVal  
      autoScalingGroupName = groupName  
      launchTemplate = templateSpecification  
    }  
  
  val groupsRequestWaiter =  
    DescribeAutoScalingGroupsRequest {  
      autoScalingGroupNames = listOf(groupName)  
    }  
  
  AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
    autoScalingClient.updateAutoScalingGroup(groupRequest)  
    autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)  
    println("You successfully updated the Auto Scaling group $groupName")  
  }  
}
```

```
    }  
  }  
  
suspend fun createAutoScalingGroup(  
    groupName: String,  
    launchTemplateNameVal: String,  
    serviceLinkedRoleARNVal: String,  
    vpcZoneIdVal: String,  
) {  
    val templateSpecification =  
        LaunchTemplateSpecification {  
            launchTemplateName = launchTemplateNameVal  
        }  
  
    val request =  
        CreateAutoScalingGroupRequest {  
            autoScalingGroupName = groupName  
            availabilityZones = listOf("us-east-1a")  
            launchTemplate = templateSpecification  
            maxSize = 1  
            minSize = 1  
            vpcZoneIdentifier = vpcZoneIdVal  
            serviceLinkedRoleArn = serviceLinkedRoleARNVal  
        }  
  
    // This object is required for the waiter call.  
    val groupsRequestWaiter =  
        DescribeAutoScalingGroupsRequest {  
            autoScalingGroupNames = listOf(groupName)  
        }  
  
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
        autoScalingClient.createAutoScalingGroup(request)  
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)  
        println("$groupName was created!")  
    }  
}  
  
suspend fun describeAutoScalingInstance(id: String) {  
    val describeAutoScalingInstancesRequest =  
        DescribeAutoScalingInstancesRequest {  
            instanceIds = listOf(id)  
        }  
}
```

```
AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
    val response =
    autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
        autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
            response.autoScalingGroups?.forEach { group ->
                println("The group name is ${group.autoScalingGroupName}")
                println("The group ARN is ${group.autoScalingGroupArn}")

                group.instances?.forEach { instance ->
                    instanceId = instance.instanceId.toString()
                }
            }
    }
    return instanceId
}
```

```
suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
            println("The max number of Auto Scaling groups is
                ${response.maxNumberOfAutoScalingGroups}")
            println("The current number of Auto Scaling groups is
                ${response.numberOfWorkingAutoScalingGroups}")
        }
    }

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
 - [CreateAutoScalingGroup](#)

- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

PHP

SDK für PHP

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;
```

```
public function runExample()
{
    echo("\n");
    echo("-----\n");
    print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
PHP!\n");
    echo("-----\n");

    $clientArgs = [
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ];
    $uniqid = uniqid();

    $this->autoScalingClient = new AutoScalingClient($clientArgs);
    $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
    $this->cloudWatchClient = new CloudWatchClient($clientArgs);

    AWSServiceClass::$waitTime = 5;
    AWSServiceClass::$maxWaitAttempts = 20;

    /**
     * Step 0: Create an EC2 launch template that you'll use to create an
    Auto Scaling group.
     */
    $this->ec2Client = new EC2Client($clientArgs);
    $this->templateName = "example_launch_template_{$uniqid}";
    $instanceType = "t1.micro";
    $amiId = "ami-0ca285d4c2cda3300";
    $launchTemplate = $this->ec2Client->createLaunchTemplate(
        [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
        ]
    );

    /**
     * Step 1: CreateAutoScalingGroup: pass it the launch template you
    created in step 0.
     */
}
```

```
    */
    $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

    $this->autoScalingGroupName = "demoAutoScalingGroupName_`$uniqid";
    $minSize = 1;
    $maxSize = 1;
    $launchTemplateId = $launchTemplate['LaunchTemplate']
['LaunchTemplateId'];
    $this->autoScalingService->createAutoScalingGroup(
        $this->autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    );

    $this->autoScalingService->waitUntilGroupInService([$this->
autoScalingGroupName]);
    $autoScalingGroup = $this->autoScalingService->
describeAutoScalingGroups([$this->autoScalingGroupName]);

    /**
     * Step 2: DescribeAutoScalingInstances: show that one instance has
     launched.
     */
    $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
    $instances = $this->autoScalingService->
describeAutoScalingInstances($instanceIds);
    echo "The Auto Scaling group {$this->autoScalingGroupName} was created
successfully.\n";
    echo count($instances['AutoScalingInstances']) . " instances were created
for the group.\n";
    echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
number of instances for the group.\n";

    /**
     * Step 3: EnableMetricsCollection: enable all metrics or a subset.
     */
    $this->autoScalingService->enableMetricsCollection($this->
autoScalingGroupName, "1Minute");

    /**
```

```
    * Step 4: UpdateAutoScalingGroup: update max size to 3.
    */
    echo "Updating the max number of instances to 3.\n";
    $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

    /**
    * Step 5: DescribeAutoScalingGroups: show the current state of the
    group.
    */
    $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
    echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
    echo " is the updated max number of instances for the group.\n";

    $limits = $this->autoScalingService->describeAccountLimits();
    echo "Here are your account limits:\n";
    echo "MaxNumberOfAutoScalingGroups:
{$limits['MaxNumberOfAutoScalingGroups']}\n";
    echo "MaxNumberOfLaunchConfigurations:
{$limits['MaxNumberOfLaunchConfigurations']}\n";
    echo "NumberOfAutoScalingGroups:
{$limits['NumberOfAutoScalingGroups']}\n";
    echo "NumberOfLaunchConfigurations:
{$limits['NumberOfLaunchConfigurations']}\n";

    /**
    * Step 6: SetDesiredCapacity: set desired capacity to 2.
    */
    $this->autoScalingService->setDesiredCapacity($this-
>autoScalingGroupName, 2);
    sleep(10); // Wait for the group to start processing the request.
    $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

    /**
    * Step 7: DescribeAutoScalingInstances: show that two instances are
    launched.
    */
    $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
    foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
        echo "There is a group named:
{$autoScalingGroup['AutoScalingGroupName']}";
```

```

        echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\\n";
        foreach ($autoScalingGroup['Instances'] as $instance) {
            echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
with id of: ";
            echo "{$instance['InstanceId']} and a lifecycle state of:
{$instance['LifecycleState']}.\\n";
        }
    }

    /**
     * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
instances in the group.
    */
    $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
    do {
        sleep(10);
        $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
    } while (count($instances['AutoScalingInstances']) > 0);
    do {
        sleep(10);
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
    } while (count($instances) < 2);
    $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
    foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
        echo "There is a group named:
{$autoScalingGroup['AutoScalingGroupName']}";
        echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\\n";
        foreach ($autoScalingGroup['Instances'] as $instance) {
            echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
with id of: ";
            echo "{$instance['InstanceId']} and a lifecycle state of:
{$instance['LifecycleState']}.\\n";
        }
    }

    /**
     * Step 9: DescribeScalingActivities: list the scaling activities that
have occurred for the group so far.
    */

```

```
    $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
    echo "We found " . count($activities['Activities']) . " activities.\n";
    foreach ($activities['Activities'] as $activity) {
        echo "{$activity['ActivityId']} - {$activity['StartTime']} -
{$activity['Description']}\n";
    }

/**
 * Step 10: Use the Amazon CloudWatch API to get and show some metrics
collected for the group.
 */
$metricsNamespace = 'AWS/AutoScaling';
$metricsDimensions = [
    [
        'Name' => 'AutoScalingGroupName',
        'Value' => $autoScalingGroup['AutoScalingGroupName'],
    ],
];
$metrics = $this->cloudWatchClient->listMetrics(
    [
        'Dimensions' => $metricsDimensions,
        'Namespace' => $metricsNamespace,
    ]
);
foreach ($metrics['Metrics'] as $metric) {
    $timespan = 5;
    if ($metric['MetricName'] != 'GroupTotalCapacity' &&
$metric['MetricName'] != 'GroupMaxSize') {
        continue;
    }
    echo "Over the last $timespan minutes, {$metric['MetricName']}
recorded:\n";
    $stats = $this->cloudWatchClient->getMetricStatistics(
        [
            'Dimensions' => $metricsDimensions,
            'EndTime' => time(),
            'StartTime' => time() - (5 * 60),
            'MetricName' => $metric['MetricName'],
            'Namespace' => $metricsNamespace,
            'Period' => 60,
            'Statistics' => ['Sum'],
        ]
    );
};
```

```
        foreach ($stats['Datapoints'] as $stat) {
            echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
        }
    }

    return $instances;
}

public function cleanUp()
{
    /**
     * Step 11: DisableMetricsCollection: disable all metrics.
     */
    $this->autoScalingService->disableMetricsCollection($this->autoScalingGroupName);

    /**
     * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
     instances.
     * - UpdateAutoScalingGroup with MinSize=0
     * - TerminateInstanceInAutoScalingGroup for each instance,
     *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
     stop.
     * - Now you can delete the group.
     */
    $this->autoScalingService->updateAutoScalingGroup($this->autoScalingGroupName, ['MinSize' => 0]);
    $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this->autoScalingGroupName);
    $this->autoScalingService->waitUntilGroupInService([$this->autoScalingGroupName]);
    $this->autoScalingService->deleteAutoScalingGroup($this->autoScalingGroupName);

    /**
     * Step 13: Delete launch template.
     */
    $this->ec2Client->deleteLaunchTemplate(
        [
            'LaunchTemplateName' => $this->templateName,
        ]
    );
}
```

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) ->
    None:
    """
        Runs the scenario demonstrating the management of Auto Scaling groups and
        instances.

        :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
        Scaling groups.
        :param svc_helper: An instance of the ServiceHelper that interacts with AWS
        services.
        :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")

    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
        instances."
    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
        new one: "
    )
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following
        specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
```

```
    template = svc_helper.create_template(template_name, inst_type, ami_id)
print("-" * 88)

print("Let's create an Auto Scaling group.")
group_name = q.ask("Enter a name for the group: ", q.non_empty)
zones = svc_helper.get_availability_zones()
print("EC2 instances can be created in the following Availability Zones:")
for index, zone in enumerate(zones):
    print(f"\t{index+1}. {zone}")
print(f"\t{len(zones)+1}. All zones")
zone_sel = q.ask(
    "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) +
1)
)
group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
print(f"Creating group {group_name}...")
as_wrapper.create_autoscaling_group(group_name, group_zones, template_name,
1, 1)
wait(10)
group = as_wrapper.describe_group(group_name)
logger.info("Created Auto Scaling group %s.", group_name)
print("Created group:")
pp(group)
print("Waiting for instance to start...")
wait_for_group(group_name, as_wrapper)
print("-" * 88)

use_metrics = q.ask(
    "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
    q.is_yn,
)
if use_metrics:
    as_wrapper.enable_metrics(
        group_name,
        [
            "GroupMinSize",
            "GroupMaxSize",
            "GroupDesiredCapacity",
            "GroupInServiceInstances",
            "GroupTotalInstances",
        ],
    )
    logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
```

```
    print(f"Metrics enabled for {group_name}.")
print("-" * 88)

print(f"Let's update the maximum number of instances in {group_name} from 1
to 3.")
q.ask("Press Enter when you're ready.")
as_wrapper.update_group(group_name, MaxSize=3)
group = as_wrapper.describe_group(group_name)
logger.info("Updated maximum size for group %s to 3.", group_name)
print("The group still has one running instance, but can have up to three:")
print_simplified_group(group)
print("-" * 88)

print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
q.ask("Press Enter when you're ready.")
as_wrapper.set_desired_capacity(group_name, 2)
wait(10)
group = as_wrapper.describe_group(group_name)
logger.info("Set desired capacity for group %s to 2.", group_name)
print("Here's the current state of the group:")
print_simplified_group(group)
print("-" * 88)
print("Waiting for the new instance to start...")
instance_ids = wait_for_group(group_name, as_wrapper)
print("-" * 88)

print(f"Let's terminate one of the instances in {group_name}.")
print("Because the desired capacity is 2, another instance will start.")
print("The currently running instances are:")
for index, inst_id in enumerate(instance_ids):
    print(f"\t{index+1}. {inst_id}")
inst_sel = q.ask(
    "Which instance do you want to stop? ",
    q.is_int,
    q.in_range(1, len(instance_ids) + 1),
)
print(f"Stopping {instance_ids[inst_sel-1]}...")
as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
wait(10)
group = as_wrapper.describe_group(group_name)
logger.info(
    "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
group_name
)
```

```

print(f"Here's the state of {group_name}:")
print_simplified_group(group)
print("Waiting for the scaling activities to complete...")
wait_for_group(group_name, as_wrapper)
print("-" * 88)

print(f"Let's get a report of scaling activities for {group_name}.")
q.ask("Press Enter when you're ready.")
activities = as_wrapper.describe_scaling_activities(group_name)
logger.info(
    "Retrieved %d scaling activities for group %s.", len(activities),
group_name
)
print(
    f"Found {len(activities)} activities.\n"
    f"Activities are ordered with the most recent one first:"
)
for act in activities:
    pp(act)
print("-" * 88)

if use_metrics:
    print("Let's look at CloudWatch metrics.")
    metric_namespace = "AWS/AutoScaling"
    metric_dimensions = [{"Name": "AutoScalingGroupName", "Value":
group_name}]
    print(f"The following metrics are enabled for {group_name}:")
    done = False
    while not done:
        metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
        for index, metric in enumerate(metrics):
            print(f"\t{index+1}. {metric.name}")
        print(f"\t{len(metrics)+1}. None")
        metric_sel = q.ask(
            "Which metric do you want to see? ",
            q.is_int,
            q.in_range(1, len(metrics) + 1),
        )
        if metric_sel < len(metrics) + 1:
            span = 5
            metric = metrics[metric_sel - 1]
            print(f"Over the last {span} minutes, {metric.name} recorded:")
            # CloudWatch metric times are in the UTC+0 time zone.
            now = datetime.now(timezone.utc)

```

```
        metric_data = svc_helper.get_metric_statistics(
            metric_dimensions, metric, now - timedelta(minutes=span), now
        )
        pp(metric_data)
        if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
            done = True
        else:
            done = True

print(f"Let's clean up.")
q.ask("Press Enter when you're ready.")
if use_metrics:
    print(f"Stopping metrics collection for {group_name}.")
    as_wrapper.disable_metrics(group_name)
    logger.info("Disabled metrics collection for group %s.", group_name)

print(
    "You must terminate all instances in the group before you can delete the
group."
)
print("Set minimum size to 0.")
as_wrapper.update_group(group_name, MinSize=0)
group = as_wrapper.describe_group(group_name)
instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
for inst_id in instance_ids:
    print(f"Stopping {inst_id}.")
    as_wrapper.terminate_instance(inst_id, True)
    logger.info("Terminated instance %s in group %s.", inst_id, group_name)
print("Waiting for instances to stop...")
wait_for_instances(instance_ids, as_wrapper)
print(f"Deleting {group_name}.")
as_wrapper.delete_autoscaling_group(group_name)
logger.info("Deleted Auto Scaling group %s.", group_name)
print("-" * 88)

if template is not None:
    if q.ask(
        f"Do you want to delete launch template {template_name} used in this
demo (y/n)? "
    ):
        svc_helper.delete_template(template_name)
        logger.info("Deleted launch template %s.", template_name)
        print("Template deleted.")
```

```

print("\nThanks for watching!")
print("-" * 88)

if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")

```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Startvorlagen und Metriken zu verwalten. Diese Funktionen umfassen Amazon EC2 und CloudWatch Aktionen.

```

class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""

    def __init__(self, ec2_client, cloudwatch_resource):
        """
        :param ec2_client: A Boto3 Amazon EC2 client.
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.ec2_client = ec2_client
        self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for
        instances
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateName=[template_name]
            )
            template = response["LaunchTemplates"][0]

```

```

        logger.info("Launch template %s retrieved successfully.",
template_name)
        return template
    except ClientError as err:
        if (
            err.response["Error"]["Code"]
            == "InvalidLaunchTemplateName.NotFoundException"
        ):
            logger.warning("Launch template %s does not exist.",
template_name)
        else:
            logger.error(
                "Couldn't verify launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
    """
    Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.

    :param template_name: The name to give to the template.
    :param inst_type: The type of the instance, such as t1.micro.
    :param ami_id: The ID of the Amazon Machine Image (AMI) to use when
creating
        an instance.
    :return: Information about the newly created template.
    :raises ClientError: If there is an error creating the launch template.
    """
    try:
        response = self.ec2_client.create_launch_template(
            LaunchTemplateName=template_name,
            LaunchTemplateData={"InstanceType": inst_type, "ImageId":
ami_id},
        )
        template = response["LaunchTemplate"]
        logger.info(
            "Created launch template %s with instance type %s and AMI ID
%s.",
            template_name,

```

```
        inst_type,
        ami_id,
    )
    return template
except ClientError as err:
    logger.error(
        "Couldn't create launch template %s. Error: %s: %s",
        template_name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

def delete_template(self, template_name: str) -> None:
    """
    Deletes a launch template.

    :param template_name: The name of the template to delete.
    :raises ClientError: If there is an error deleting the launch template.
    """
    try:
self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
        logger.info("Deleted launch template %s.", template_name)
    except ClientError as err:
        logger.error(
            "Couldn't delete launch template %s. Error: %s: %s",
            template_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def get_availability_zones(self) -> list:
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    :raises ClientError: If there is an error retrieving availability zones.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
```

```
        logger.info("Retrieved availability zones: %s.", ", ".join(zones))
        return zones
    except ClientError as err:
        logger.error(
            "Couldn't get availability zones. Error: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def get_metrics(self, namespace: str, dimensions: list) -> list:
    """
    Gets a list of CloudWatch metrics filtered by namespace and dimensions.

    :param namespace: The namespace of the metrics to look up.
    :param dimensions: The dimensions of the metrics to look up.
    :return: The list of metrics.
    :raises ClientError: If there is an error retrieving CloudWatch metrics.
    """
    try:
        metrics = list(
            self.cloudwatch_resource.metrics.filter(
                Namespace=namespace, Dimensions=dimensions
            )
        )
        logger.info(
            "Retrieved metrics for namespace %s with dimensions %s.",
            namespace,
            dimensions,
        )
        return metrics
    except ClientError as err:
        logger.error(
            "Couldn't get metrics for %s, %s. Error: %s: %s",
            namespace,
            dimensions,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

    @staticmethod
    def get_metric_statistics(
        dimensions: list, metric, start: datetime, end: datetime
```

```

) -> list:
    """
    Gets statistics for a CloudWatch metric within a specified time span.

    :param dimensions: The dimensions of the metric.
    :param metric: The metric to look up.
    :param start: The start of the time span for retrieved metrics.
    :param end: The end of the time span for retrieved metrics.
    :return: The list of data points found for the specified metric.
    :raises ClientError: If there is an error retrieving metric statistics.
    """
    try:
        response = metric.get_statistics(
            Dimensions=dimensions,
            StartTime=start,
            EndTime=end,
            Period=60,
            Statistics=["Sum"],
        )
        data = response["Datapoints"]
        logger.info("Retrieved statistics for metric %s.", metric.name)
        return data
    except ClientError as err:
        logger.error(
            "Couldn't get statistics for metric %s. Error: %s: %s",
            metric.name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def print_simplified_group(group: dict) -> None:
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
        {group['DesiredCapacity']}"
    )

```

```
)
if group["Instances"]:
    print(f"\tInstances:")
    for inst in group["Instances"]:
        print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")

def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)

def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) ->
list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
    """
    ready = False
    instances = []
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids
else []
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
```

```
        f"Here are the details of the instance{'s' if len(instances) > 1 else  
    ''}:"  
    )  
    for instance in instances:  
        pp(instance)  
    return instance_ids
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
[package]  
name = "autoscaling-code-examples"  
version = "0.1.0"
```

```
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
  <dpsouth@amazon.com>"]
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
clap = { version = "4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"

use std::{collections::BTreeSet, fmt::Display};

use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}
```

```

}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:");
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "");
        }
        Ok(())
    }
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();

    // 1. Create an EC2 launch template that you'll use to create an auto scaling
    group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
    template.
    // 2. CreateAutoScalingGroup: pass it the launch template you created in step
    0. Give it min/max of 1 instance.
    // 4. EnableMetricsCollection: enable all metrics or a subset.
    let scenario = match
    AutoScalingScenario::prepare_scenario(&shared_config).await {
        Ok(scenario) => scenario,
        Err(errs) => {
            let err_str = errs
                .into_iter()
                .map(|e| e.to_string())
                .collect::<Vec<String>>()
                .join(", ");
            return Err(anyhow!("Failed to initialize scenario: {err_str}"));
        }
    };

    info!("Prepared autoscaling scenario:\n{scenario}");

    let stable = scenario.wait_for_stable(1).await;
    if let Err(err) = stable {
        warnings.push(

```

```
        "There was a problem while waiting for group to be stable",
        err,
    );
}

// 3. DescribeAutoScalingInstances: show that one instance has launched.
show_scenario_description(
    &scenario,
    "show that the group was created and one instance has launched",
)
.await;

// 5. UpdateAutoScalingGroup: update max size to 3.
let scale_max_size = scenario.scale_max_size(3).await;
if let Err(err) = scale_max_size {
    warnings.push("There was a problem scaling max size", err);
}

// 6. DescribeAutoScalingGroups: the current state of the group
show_scenario_description(
    &scenario,
    "show the current state of the group after setting max size",
)
.await;

// 7. SetDesiredCapacity: set desired capacity to 2.
let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
if let Err(err) = scale_desired_capacity {
    warnings.push("There was a problem setting desired capacity", err);
}

// Wait for a second instance to launch.
let stable = scenario.wait_for_stable(2).await;
if let Err(err) = stable {
    warnings.push(
        "There was a problem while waiting for group to be stable",
        err,
    );
}

// 8. DescribeAutoScalingInstances: show that two instances are launched.
show_scenario_description(
    &scenario,
    "show that two instances are launched after setting desired capacity",
```

```
)
.await;

let ids_before = scenario
  .list_instances()
  .await
  .map(|v| v.into_iter().collect::<BTreeSet<_>>())
  .unwrap_or_default();

// 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in
the group.
let terminate_some_instance = scenario.terminate_some_instance().await;
if let Err(err) = terminate_some_instance {
  warnings.push("There was a problem replacing an instance", err);
}

let wait_after_terminate = scenario.wait_for_stable(1).await;
if let Err(err) = wait_after_terminate {
  warnings.push(
    "There was a problem waiting after terminating an instance",
    err,
  );
}

let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
if let Err(err) = wait_scale_up_after_terminate {
  warnings.push(
    "There was a problem waiting for scale up after terminating an
instance",
    err,
  );
}

let ids_after = scenario
  .list_instances()
  .await
  .map(|v| v.into_iter().collect::<BTreeSet<_>>())
  .unwrap_or_default();

let difference = ids_after.intersection(&ids_before).count();
if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
  warnings.push(
    "Before and after set not different",
    ScenarioError::with(format!("{difference}")),
  );
}
```

```
    );
}

// 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
show_scenario_description(
    &scenario,
    "list the scaling activities that have occurred for the group so far",
)
.await;

// 11. DisableMetricsCollection
let scale_group = scenario.scale_group_to_zero().await;
if let Err(err) = scale_group {
    warnings.push("There was a problem scaling the group to 0", err);
}
show_scenario_description(&scenario, "Scenario scaled to 0").await;

// 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
// 13. Delete LaunchTemplate.
let clean_scenario = scenario.clean_scenario().await;
if let Err(errs) = clean_scenario {
    for err in errs {
        warnings.push("There was a problem cleaning the scenario", err);
    }
} else {
    info!("The scenario has been cleaned up!");
}

if warnings.is_empty() {
    Ok(())
} else {
    Err(anyhow!(
        "There were warnings during scenario execution:\n{warnings}"
    ))
}
}

pub mod scenario;

use std::{
    error::Error,
```

```
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
    "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
const AUTOSCALING_GROUP_NAME: &str =
    "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25
    seconds.
const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at
    a time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
            ))
        }
    }
}
```

```
    } else {
        tokio::time::sleep(WAIT_TIME).await;
        Ok(())
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\n",
            self.launch_template_arn
        ))?;
        f.write_fmt(format_args!(
            "\tScaling Group Name: {}\n",
            self.auto_scaling_group_name
        ))?;

        Ok(())
    }
}

pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t\t\tGroup status:");
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t\t- {status}");
                }
            }
        }
    }
}
```

```

        Err(e) => writeln!(f, "\t\t! - {e}")?,
    }
    writeln!(f, "\t\t\t\t\tInstances:");
    match &self.instances {
        Ok(instances) => {
            for instance in instances {
                writeln!(f, "\t\t\t- {instance}")?;
            }
        }
        Err(e) => writeln!(f, "\t\t\t! {e}")?,
    }

    writeln!(f, "\t\t\t\t\tActivities:");
    match &self.activities {
        Ok(activities) => {
            for activity in activities {
                writeln!(
                    f,
                    "\t\t\t- {} Progress: {}% Status: {:?} End: {:?}",
                    activity.cause().unwrap_or("Unknown"),
                    activity.progress.unwrap_or(-1),
                    activity.status_code(),
                    // activity.status_message().unwrap_or_default()
                    activity.end_time(),
                )?;
            }
        }
        Err(e) => writeln!(f, "\t\t\t! {e}")?,
    }

    Ok(())
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(|s| s.to_string()),

```

```

        code: err.code().map(|s| s.to_string()),
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("{code}"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
    Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
    // for this example

```

```

    // the ScenarioError captures the underlying information in MetadataError and
    // treats it as a
    // single Error from this Crate. In other contexts, it may be appropriate to
    // model the error
    // as including the SdkError as its source.
}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
    Vec<ScenarioError>> {
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);

        let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

        // Before creating any resources, prepare the list of AZs
        let availability_zones = ec2.describe_availability_zones().send().await;
        if let Err(err) = availability_zones {
            return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
        }

        let availability_zones: Vec<String> = availability_zones
            .unwrap()
            .availability_zones
            .unwrap_or_default()
            .iter()
            .take(3)
            .map(|z| z.zone_name.clone().unwrap())
            .collect();

        // 1. Create an EC2 launch template that you'll use to create an auto
        // scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
        // template.
        // * Recommended: InstanceType='t1.micro',
        // ImageId='ami-0ca285d4c2cda3300'
        let create_launch_template = ec2

```

```

        .create_launch_template()
        .launch_template_name(LAUNCH_TEMPLATE_NAME)
        .launch_template_data(
            RequestLaunchTemplateData::builder()
                .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
                .image_id("ami-0ca285d4c2cda3300")
                .build(),
        )
        .send()
        .await
        .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)])?;

        let launch_template_arn = match create_launch_template.launch_template {
            Some(launch_template) =>
                launch_template.launch_template_id.unwrap_or_default(),
            None => {
                // Try to delete the launch template
                let _ = ec2
                    .delete_launch_template()
                    .launch_template_name(LAUNCH_TEMPLATE_NAME)
                    .send()
                    .await;
                return Err(vec![ScenarioError::with("Failed to load launch
template")]);
            }
        };

        // 2. CreateAutoScalingGroup: pass it the launch template you created in
        // step 0. Give it min/max of 1 instance.
        // You can use EC2.describe_availability_zones() to get a list of AZs
        // (you have to specify an AZ when you create the group).
        // Wait for instance to launch. Use a waiter if you have one, otherwise
        DescribeAutoScalingInstances until LifecycleState='InService'
        if let Err(err) = autoscaling
            .create_auto_scaling_group()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .launch_template(
                LaunchTemplateSpecification::builder()
                    .launch_template_id(launch_template_arn.clone())
                    .version("$Latest")
                    .build(),
            )
            .max_size(1)

```

```
.min_size(1)
.set_availability_zones(Some(availability_zones))
.send()
.await
{
let mut errs = vec![ScenarioError::new(
    "Failed to create autoscaling group",
    &err,
)];

if let Err(err) = autoscaling
    .delete_auto_scaling_group()
    .auto_scaling_group_name(auto_scaling_group_name.as_str())
    .send()
    .await
{
    errs.push(ScenarioError::new(
        "Failed to clean up autoscaling group",
        &err,
    ));
}

if let Err(err) = ec2
    .delete_launch_template()
    .launch_template_id(launch_template_arn.clone())
    .send()
    .await
{
    errs.push(ScenarioError::new(
        "Failed to clean up launch template",
        &err,
    ));
}
return Err(errs);
}

let scenario = AutoScalingScenario {
    ec2,
    autoscaling: autoscaling.clone(), // Clients are cheap so cloning
here to prevent a move is ok.
    auto_scaling_group_name: auto_scaling_group_name.clone(),
    launch_template_arn,
};
```

```
let enable_metrics_collection = autoscaling
    .enable_metrics_collection()
    .auto_scaling_group_name(auto_scaling_group_name.as_str())
    .granularity("1Minute")
    .set_metrics(Some(vec![
        String::from("GroupMinSize"),
        String::from("GroupMaxSize"),
        String::from("GroupDesiredCapacity"),
        String::from("GroupInServiceInstances"),
        String::from("GroupTotalInstances"),
    ]))
    .send()
    .await;

match enable_metrics_collection {
    Ok(_) => Ok(scenario),
    Err(err) => {
        scenario.clean_scenario().await?;
        Err(vec![ScenarioError::new(
            "Failed to enable metrics collections for group",
            &err,
        )])
    }
}

pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
    let _ = self.wait_for_no_scaling().await;
    let delete_group = self
        .autoscaling
        .delete_auto_scaling_group()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .send()
        .await;

    // 14. Delete LaunchTemplate.
    let delete_launch_template = self
        .ec2
        .delete_launch_template()
        .launch_template_id(self.launch_template_arn.clone())
        .send()
        .await;

    let early_exit = match (delete_group, delete_launch_template) {
```

```

        (Ok(_), Ok(_)) => Ok(()),
        (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
            "There was an error cleaning the launch template",
            &e,
        )]),
        (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
            "There was an error cleaning the scale group",
            &e,
        )]),
        (Err(e1), Err(e2)) => Err(vec![
            ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
            ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
        ]),
    };

    if early_exit.is_err() {
        early_exit
    } else {
        // Wait for delete_group to finish
        let waiter = Waiter::new();
        let mut errors = Vec::<ScenarioError>::new();
        while errors.len() < 3 {
            if let Err(e) = waiter.sleep().await {
                errors.push(e);
                continue;
            }
            let describe_group = self
                .autoscaling
                .describe_auto_scaling_groups()

.auto_scaling_group_names(self.auto_scaling_group_name.clone())
                .send()
                .await;
            match describe_group {
                Ok(group) => match group.auto_scaling_groups().first() {
                    Some(group) => {
                        if group.status() != Some("Delete in progress") {
                            errors.push(ScenarioError::with(format!(
                                "Group in an unknown state while deleting:
{}",
                                group.status().unwrap_or("unknown error")
                            )));
                        }
                    }
                    None => {}
                }
            }
        }
    }
}

```

```

        return Err(errors);
    }
}
None => return Ok(()),
},
Err(err) => {
    errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
}
}
if errors.len() > 3 {
    return Err(errors);
}
}
Err(vec![ScenarioError::with(
    "Exited cleanup wait loop without retuning success or failing
after three rounds",
)])
}
}

pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
    let group = self
        .autoscaling
        .describe_auto_scaling_groups()
        .auto_scaling_group_names(self.auto_scaling_group_name.clone())
        .send()
        .await
        .map(|s| {
            s.auto_scaling_groups()
                .iter()
                .map(|s| {
                    format!(
                        "{}: {}",
                        s.auto_scaling_group_name().unwrap_or("Unknown"),
                        s.status().unwrap_or("Unknown")
                    )
                })
                .collect:::<Vec<String>>()
        })
        .map_err(|e| {
            ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
        });
}

```

```

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
        occurred for the group so far.
        // Bonus: use CloudWatch API to get and show some metrics collected for
        the group.
        // CW.ListMetrics with Namespace='AWS/AutoScaling' and
        Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        // CW.GetMetricStatistics with Statistics='Sum'. Start and End times
        must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::

```

```
if let Err(err) = describe_auto_scaling_groups {
    return Err(ScenarioError::new(
        format!(
            "Failed to get status of autoscaling group {}",
            self.auto_scaling_group_name.clone()
        )
        .as_str(),
        &err,
    ));
}

let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
let auto_scaling_group = auto_scaling_groups.first();

if auto_scaling_group.is_none() {
    return Err(ScenarioError::with(format!(
        "Could not find autoscaling group {}",
        self.auto_scaling_group_name.clone()
    )));
}

Ok(auto_scaling_group.unwrap().clone())
}

pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
    let waiter = Waiter::new();
    let mut scaling = true;
    while scaling {
        waiter.sleep().await?;
        let describe_activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map_err(|e| {
                ScenarioError::new("Failed to get autoscaling activities for
group", &e)
            })?;
        let activities = describe_activities.activities();
    }
}
```

```

        trace!(
            "Waiting for no scaling found {} activities",
            activities.len()
        );
        scaling = activities.iter().any(|a| a.progress() < Some(100));
    }
    Ok(())
}

pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError>
{
    self.wait_for_no_scaling().await?;

    let mut group = self.get_group().await?;
    let mut count = count_group_instances(&group);

    let waiter = Waiter::new();
    while count != size {
        trace!("Waiting for stable {size} (current: {count})");
        waiter.sleep().await?;
        group = self.get_group().await?;
        count = count_group_instances(&group);
    }

    Ok(())
}

pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
    // The direct way to list instances is by using
    DescribeAutoScalingGroup's instances property. However, this returns a
    Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
    // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

    // Alternatively, and for the sake of example,
    DescribeAutoScalingInstances returns a list that can be filtered by the client.
    self.autoscaling
        .describe_auto_scaling_instances()
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await
}

```

```

        .map(|items| {
            items
                .into_iter()
                .filter(|i| {
                    i.auto_scaling_group_name.as_deref()
                        == Some(self.auto_scaling_group_name.as_str())
                })
                .map(|i| i.instance_id.unwrap_or_default())
                .filter(|id| !id.is_empty())
                .collect::<Vec<String>>()
        })
        .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }

pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
    let update_group = self
        .autoscaling
        .update_auto_scaling_group()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .min_size(size)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            format!("Failer to update group to min size ({size}))").as_str(),
            &err,
        ));
    }
    Ok(())
}

pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
    // 5. UpdateAutoScalingGroup: update max size to 3.
    let update_group = self
        .autoscaling
        .update_auto_scaling_group()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .max_size(size)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            format!("Failed to update group to max size ({size}))").as_str(),

```

```
        &err,
    ));
}
Ok(())
}

pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
    // 7. SetDesiredCapacity: set desired capacity to 2.
    // Wait for a second instance to launch.
    let update_group = self
        .autoscaling
        .set_desired_capacity()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .desired_capacity(capacity)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            format!("Failed to update group to desired capacity
({capacity}))").as_str(),
            &err,
        ));
    }
    Ok(())
}

pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
    // If this fails it's fine, just means there are extra cloudwatch metrics
    events for the scale-down.
    let _ = self
        .autoscaling
        .disable_metrics_collection()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .send()
        .await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all
    instances):
    // UpdateAutoScalingGroup with MinSize=0
    let update_group = self
        .autoscaling
        .update_auto_scaling_group()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
```

```
        .min_size(0)
        .desired_capacity(0)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            "Failed to update group for scaling down&",
            &err,
        ));
    }

    let stable = self.wait_for_stable(0).await;
    if let Err(err) = stable {
        return Err(ScenarioError::with(format!(
            "Error while waiting for group to be stable on scale down: {err}"
        )));
    }

    Ok(())
}

pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
    // Retrieve a list of instances in the auto scaling group.
    let auto_scaling_group = self.get_group().await?;
    let instances = auto_scaling_group.instances();
    // Or use other logic to find an instance to terminate.
    let instance = instances.first();
    if let Some(instance) = instance {
        let instance_id = if let Some(instance_id) = instance.instance_id() {
            instance_id
        } else {
            return Err(ScenarioError::with("Missing instance id"));
        };
        let termination = self
            .ec2
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await;
        if let Err(err) = termination {
            return Err(ScenarioError::new(
                "There was a problem terminating an instance",
                &err,
            ))
        }
    }
}
```

```
        } else {
            Ok(())
        }
    } else {
        Err(ScenarioError::with("There was no instance to terminate"))
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS-SDK für Rust.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen für Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie einzelne Auto Scaling Scaling-Aktionen mit ausführen AWS SDKs. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Auto-Scaling-API auf und sind Codeausschnitte aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter [Szenarien für Auto Scaling mit AWS SDKs](#) anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der [API-Referenz zu Amazon EC2 Auto Scaling](#).

Beispiele

- [Verwendung von AttachInstances mit einer CLI](#)
- [Verwendung AttachLoadBalancerTargetGroups mit einem AWS SDK oder CLI](#)
- [Verwendung von AttachLoadBalancers mit einer CLI](#)
- [Verwendung von CompleteLifecycleAction mit einer CLI](#)
- [Verwendung CreateAutoScalingGroup mit einem AWS SDK oder CLI](#)
- [Verwendung von CreateLaunchConfiguration mit einer CLI](#)
- [Verwendung von CreateOrUpdateTags mit einer CLI](#)
- [Verwendung DeleteAutoScalingGroup mit einem AWS SDK oder CLI](#)
- [Verwendung von DeleteLaunchConfiguration mit einer CLI](#)
- [Verwendung von DeleteLifecycleHook mit einer CLI](#)
- [Verwendung von DeleteNotificationConfiguration mit einer CLI](#)
- [Verwendung von DeletePolicy mit einer CLI](#)
- [Verwendung von DeleteScheduledAction mit einer CLI](#)
- [Verwendung von DeleteTags mit einer CLI](#)
- [Verwendung von DescribeAccountLimits mit einer CLI](#)
- [Verwendung von DescribeAdjustmentTypes mit einer CLI](#)
- [Verwendung DescribeAutoScalingGroups mit einem AWS SDK oder CLI](#)
- [Verwendung DescribeAutoScalingInstances mit einem AWS SDK oder CLI](#)
- [Verwendung von DescribeAutoScalingNotificationTypes mit einer CLI](#)
- [Verwendung von DescribeLaunchConfigurations mit einer CLI](#)
- [Verwendung von DescribeLifecycleHookTypes mit einer CLI](#)
- [Verwendung von DescribeLifecycleHooks mit einer CLI](#)
- [Verwendung von DescribeLoadBalancers mit einer CLI](#)
- [Verwendung von DescribeMetricCollectionTypes mit einer CLI](#)

- [Verwendung von DescribeNotificationConfigurations mit einer CLI](#)
- [Verwendung von DescribePolicies mit einer CLI](#)
- [Verwendung DescribeScalingActivities mit einem AWS SDK oder CLI](#)
- [Verwendung von DescribeScalingProcessTypes mit einer CLI](#)
- [Verwendung von DescribeScheduledActions mit einer CLI](#)
- [Verwendung von DescribeTags mit einer CLI](#)
- [Verwendung von DescribeTerminationPolicyTypes mit einer CLI](#)
- [Verwendung von DetachInstances mit einer CLI](#)
- [Verwendung von DetachLoadBalancers mit einer CLI](#)
- [Verwendung DisableMetricsCollection mit einem AWS SDK oder CLI](#)
- [Verwendung EnableMetricsCollection mit einem AWS SDK oder CLI](#)
- [Verwendung von EnterStandby mit einer CLI](#)
- [Verwendung von ExecutePolicy mit einer CLI](#)
- [Verwendung von ExitStandby mit einer CLI](#)
- [Verwendung von PutLifecycleHook mit einer CLI](#)
- [Verwendung von PutNotificationConfiguration mit einer CLI](#)
- [Verwendung von PutScalingPolicy mit einer CLI](#)
- [Verwendung von PutScheduledUpdateGroupAction mit einer CLI](#)
- [Verwendung von RecordLifecycleActionHeartbeat mit einer CLI](#)
- [Verwendung von ResumeProcesses mit einer CLI](#)
- [Verwendung SetDesiredCapacity mit einem AWS SDK oder CLI](#)
- [Verwendung von SetInstanceHealth mit einer CLI](#)
- [Verwendung von SetInstanceProtection mit einer CLI](#)
- [Verwendung von SuspendProcesses mit einer CLI](#)
- [Verwendung TerminateInstanceInAutoScalingGroup mit einem AWS SDK oder CLI](#)
- [Verwendung UpdateAutoScalingGroup mit einem AWS SDK oder CLI](#)

Verwendung von **AttachInstances** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie AttachInstances verwendet wird.

CLI

AWS CLI

So fügen Sie einer Auto-Scaling-Gruppe eine Instance hinzu

In diesem Beispiel wird die angegebene Instance an die spezifizierte Auto-Scaling-Gruppe angehängt.

```
aws autoscaling attach-instances \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [AttachInstances](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance an die spezifizierte Auto-Scaling-Gruppe hinzugefügt. Auto Scaling erhöht automatisch die gewünschte Kapazität der Auto-Scaling-Gruppe.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [AttachInstances AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance an die spezifizierte Auto-Scaling-Gruppe hinzugefügt. Auto Scaling erhöht automatisch die gewünschte Kapazität der Auto-Scaling-Gruppe.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [AttachInstances AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **AttachLoadBalancerTargetGroups** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AttachLoadBalancerTargetGroups` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
/// The
/// </summary>
/// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
/// <param name="targetGroupArn">The Arn for the target group.</param>
/// <returns>Async task.</returns>
public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
string targetGroupArn)
{
    await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
        new AttachLoadBalancerTargetGroupsRequest()
        {
            AutoScalingGroupName = autoScalingGroupName,
            TargetGroupARNs = new List<string>() { targetGroupArn }
        });
}
```

```
}
```

- Einzelheiten zur API finden Sie [AttachLoadBalancerTargetGroups](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So fügen Sie einer Auto-Scaling-Gruppe eine Zielgruppe hinzu

In diesem Beispiel wird die angegebene Zielgruppe zur spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling attach-load-balancer-target-groups \  
  --auto-scaling-group-name my-asg \  
  --target-group-arns arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Elastic Load Balancing und Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [AttachLoadBalancerTargetGroups](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
const client = new AutoScalingClient({});  
await client.send(  

```

```
new AttachLoadBalancerTargetGroupsCommand({
  AutoScalingGroupName: NAMES.autoScalingGroupName,
  TargetGroupARNs: [state.targetGroupArn],
}),
);
```

- Einzelheiten zur API finden Sie [AttachLoadBalancerTargetGroups](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
            created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
```

```

        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
        Scaling group.
        The target group specifies how the load balancer forwards requests to the
        instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:

```

```

        self.autoscaling_client.attach_load_balancer_target_groups(
            AutoScalingGroupName=self.group_name,
            TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
        )
        log.info(
            "Attached load balancer target group %s to auto scaling group
%s.",
            lb_target_group["TargetGroupName"],
            self.group_name,
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to attach load balancer target group
'{{lb_target_group['TargetGroupName']}}'."
        )
        if error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the resource."
            )
        elif error_code == "ServiceLinkedRoleFailure":
            log.error(
                "The operation failed because the service-linked role is not
ready or does not exist. "
                "Check that the service-linked role exists and is correctly
configured."
            )
        log.error(f"Full error:\n\t{{err}}")

```

- Einzelheiten zur API finden Sie [AttachLoadBalancerTargetGroups](#) in AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **AttachLoadBalancers** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachLoadBalancers` verwendet wird.

CLI

AWS CLI

So fügen Sie einer Auto-Scaling-Gruppe einen Classic Load Balancer hinzu

In diesem Beispiel wird der angegebene Classic Load Balancer der spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling attach-load-balancers \  
  --load-balancer-names my-load-balancer \  
  --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Elastic Load Balancing und Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [AttachLoadBalancers](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer an die spezifizierte Auto-Scaling-Gruppe hinzugefügt.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [AttachLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz \(V4\)](#).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer an die spezifizierte Auto-Scaling-Gruppe hinzugefügt.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [AttachLoadBalancers AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **CompleteLifecycleAction** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CompleteLifecycleAction` verwendet wird.

CLI

AWS CLI

So schließen Sie eine Lebenszyklusaktion ab

In diesem Beispiel wird Amazon EC2 Auto Scaling darüber informiert, dass die angegebene Lebenszyklusaktion abgeschlossen ist, sodass das Starten oder Beenden der Instance abgeschlossen werden kann.

```
aws autoscaling complete-lifecycle-action \  
  --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-action-result CONTINUE \  
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [CompleteLifecycleAction](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Lebenszyklusaktion abgeschlossen.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -  
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter [CompleteLifecycleAction AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Lebenszyklusaktion abgeschlossen.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -  
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter [CompleteLifecycleAction AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **CreateAutoScalingGroup** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAutoScalingGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
    {
        availabilityZone,
    };

    var request = new CreateAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        AvailabilityZones = zoneList,
        LaunchTemplate = templateSpecification,
        MaxSize = 6,
        MinSize = 1
    };
};
```

```
    try
    {
        var response = await
        _amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AlreadyExistsException)
    {
        Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
        return true;
    }
}
```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);
Aws::Vector<Aws::String> availabilityGroupZones;
availabilityGroupZones.push_back(
    availabilityZones[availabilityZoneChoice - 1].GetZoneName());
request.SetAvailabilityZones(availabilityGroupZones);
```

```
request.SetMaxSize(1);
request.SetMinSize(1);

Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
launchTemplateSpecification.SetLaunchTemplateName(templateName);
request.SetLaunchTemplate(launchTemplateSpecification);

Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
    autoScalingClient.CreateAutoScalingGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "Created Auto Scaling group '" << groupName << "'..."
              << std::endl;
}
else if (outcome.GetError().GetErrorType() ==
        Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
    std::cout << "Auto Scaling group '" << groupName << "' already
exists."
              << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
}
```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Auto-Scaling-Gruppe

Im folgenden Beispiel für `create-auto-scaling-group` wird eine Auto-Scaling-Gruppe in Subnetzen in mehreren Availability Zones innerhalb einer Region erstellt. Die Instances werden mit der Standardversion der angegebenen Startvorlage gestartet. Bitte beachten

Sie, dass für die meisten anderen Einstellungen, wie z. B. die Beendigungsrichtlinien und die Konfiguration der Zustandsprüfung, die Standardeinstellungen verwendet werden.

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12 \  
  --min-size 1 \  
  --max-size 5 \  
  --vpc-zone-identifizier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 2: So fügen Sie einen Application Load Balancer, einen Network Load Balancer oder einen Gateway Load Balancer hinzu

Dieses Beispiel gibt den ARN einer Zielgruppe für einen Load Balancer an, der den erwarteten Datenverkehr unterstützt. Der Typ der Zustandsprüfung gibt ELB an, sodass die Auto-Scaling-Gruppe eine Instance ersetzt, wenn Elastic Load Balancing diese als fehlerhaft meldet. Der Befehl definiert außerdem eine Wartefrist für die Zustandsprüfung von 600 Sekunden. Die Wartefrist trägt dazu bei, ein vorzeitiges Beenden neu gestarteter Instances zu verhindern.

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12 \  
  --target-group-arns arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-targets/943f017f100becff \  
  --health-check-type ELB \  
  --health-check-grace-period 600 \  
  --min-size 1 \  
  --max-size 5 \  
  --vpc-zone-identifizier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Elastic Load Balancing und Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So geben Sie eine Platzierungsgruppe an und verwenden die neueste Version der Startvorlage

Dieses Beispiel startet Instances in einer Platzierungsgruppe innerhalb einer einzelnen Availability Zone. Dies kann für Gruppen mit niedriger Latenz und HPC-Workloads nützlich sein. Außerdem werden in diesem Beispiel die minimale Größe, die maximale Größe und die gewünschte Kapazität der Gruppe angegeben.

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest' \  
  --min-size 1 \  
  --max-size 5 \  
  --desired-capacity 3 \  
  --placement-group my-placement-group \  
  --vpc-zone-identifier "subnet-6194ea3b"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Platzierungsgruppen](#) im Benutzerhandbuch für Amazon EC2 für Linux-Instances.

Beispiel 4: So geben Sie eine Auto-Scaling-Gruppe mit einer einzelnen Instance an und verwenden eine bestimmte Version der Startvorlage

Dieses Beispiel erstellt eine Auto-Scaling-Gruppe mit einer minimalen und maximalen Kapazität von 1, um sicherzustellen, dass eine Instance ausgeführt wird. Der Befehl gibt außerdem Version 1 einer Startvorlage an, in der die ID einer vorhandenen ENI angegeben ist. Wenn Sie eine Startvorlage verwenden, die eine vorhandene ENI für eth0 angibt, müssen Sie für die Auto-Scaling-Gruppe eine Availability Zone angeben, die mit der Netzwerkschnittstelle übereinstimmt, ohne in der Anforderung zusätzlich eine Subnetz-ID anzugeben.

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg-single-instance \  
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1' \  
  --min-size 1 \  
  --max-size 1 \  
  --availability-zones us-west-2a
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 5: So geben Sie eine andere Beendigungsrichtlinie an

In diesem Beispiel wird eine Auto-Scaling-Gruppe mithilfe einer Startkonfiguration erstellt und die Beendigungsrichtlinie so festgelegt, dass die ältesten Instances zuerst beendet werden. Der Befehl weist außerdem der Gruppe und ihren Instances ein Tag mit dem Schlüssel `Role` und dem Wert `WebServer` zu.

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-configuration-name my-lc \  
  --min-size 1 \  
  --max-size 5 \  
  --termination-policies "OldestInstance" \  
  --tags "ResourceId=my-asg,ResourceType=auto-scaling-  
group,Key=Role,Value=WebServer,PropagateAtLaunch=true" \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Arbeiten mit Beendigungsrichtlinien für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 6: So geben Sie einen Start-Lebenszyklus-Hook an

Das folgende Beispiel erstellt eine Auto-Scaling-Gruppe mit einem Lebenszyklus-Hook, der eine benutzerdefinierte Aktion beim Starten einer Instance unterstützt.

```
aws autoscaling create-auto-scaling-group \  
  --cli-input-json file://~/config.json
```

Inhalt der `config.json`-Datei:

```
{  
  "AutoScalingGroupName": "my-asg",  
  "LaunchTemplate": {  
    "LaunchTemplateId": "lt-1234567890abcde12"  
  },  
  "LifecycleHookSpecificationList": [{  
    "LifecycleHookName": "my-launch-hook",  
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",  
    "NotificationTargetARN": "arn:aws:sqs:us-west-2:123456789012:my-sqs-  
queue",
```

```

    "RoleARN": "arn:aws:iam::123456789012:role/my-notification-role",
    "NotificationMetadata": "SQS message metadata",
    "HeartbeatTimeout": 4800,
    "DefaultResult": "ABANDON"
  }],
  "MinSize": 1,
  "MaxSize": 5,
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags": [{
    "ResourceType": "auto-scaling-group",
    "ResourceId": "my-asg",
    "PropagateAtLaunch": true,
    "Value": "test",
    "Key": "environment"
  }]
}

```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 7: So geben Sie einen Beendigungs-Lebenszyklus-Hook an

Das folgende Beispiel erstellt eine Auto-Scaling-Gruppe mit einem Lebenszyklus-Hook, der eine benutzerdefinierte Aktion beim Beenden einer Instance unterstützt.

```

aws autoscaling create-auto-scaling-group \
  --cli-input-json file://~/config.json

```

Inhalt von config.json:

```

{
  "AutoScalingGroupName": "my-asg",
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-1234567890abcde12"
  },
  "LifecycleHookSpecificationList": [{
    "LifecycleHookName": "my-termination-hook",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "HeartbeatTimeout": 120,
    "DefaultResult": "CONTINUE"
  }]
}

```

```
    ]],  
    "MinSize": 1,  
    "MaxSize": 5,  
    "TargetGroupARNs": [  
        "arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-  
targets/73e2d6bc24d8a067"  
    ],  
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"  
}
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 8: So geben Sie eine benutzerdefinierte Beendigungsrichtlinie an

In diesem Beispiel wird eine Auto-Scaling-Gruppe erstellt, die eine benutzerdefinierte Lambda-Funktionsrichtlinie für Beendigungen angibt, die Amazon EC2 Auto Scaling mitteilt, welche Instances sicher beim Abskalieren beendet werden können.

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg-single-instance \  
  --launch-template LaunchTemplateName=my-template-for-auto-scaling \  
  --min-size 1 \  
  --max-size 5 \  
  --termination-policies "arn:aws:lambda:us-  
west-2:123456789012:function>HelloFunction:prod" \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Erstellen einer benutzerdefinierten Beendigungsrichtlinie mit Lambda](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <groupName> <launchTemplateName> <serviceLinkedRoleARN>
                <vpcZoneId>

                Where:
```

```
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.\s
        vpcZoneId - A subnet Id for a virtual private cloud (VPC)
where instances in the Auto Scaling group can be created.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String groupName = args[0];
    String launchTemplateName = args[1];
    String vpcZoneId = args[2];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
    autoScalingClient.close();
}

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
        String groupName,
        String launchTemplateName,
        String vpcZoneId) {

    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
```

```

        .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {

```

```
val templateSpecification =
    LaunchTemplateSpecification {
        launchTemplateName = launchTemplateNameVal
    }

val request =
    CreateAutoScalingGroupRequest {
        autoScalingGroupName = groupName
        availabilityZones = listOf("us-east-1a")
        launchTemplate = templateSpecification
        maxSize = 1
        minSize = 1
        vpcZoneIdentifier = vpcZoneIdVal
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
    }

// This object is required for the waiter call.
val groupsRequestWaiter =
    DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
    autoScalingClient.createAutoScalingGroup(request)
    autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
    println("$groupName was created!")
}
}
```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function createAutoScalingGroup(
    $autoScalingGroupName,
    $availabilityZones,
    $minSize,
    $maxSize,
    $launchTemplateId
) {
    return $this->autoScalingClient->createAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'AvailabilityZones' => $availabilityZones,
        'MinSize' => $minSize,
        'MaxSize' => $maxSize,
        'LaunchTemplate' => [
            'LaunchTemplateId' => $launchTemplateId,
        ],
    ]);
}
```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Auto-Scaling-Gruppe mit dem angegebenen Namen und den angegebenen Attributen erstellt. Die standardmäßige gewünschte Kapazität entspricht der minimalen Größe. Daher startet diese Auto-Scaling-Gruppe zwei Instances – jeweils eine in jeder der angegebenen beiden Availability Zones.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-
lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Einzelheiten zur API finden Sie unter [CreateAutoScalingGroup AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Auto-Scaling-Gruppe mit dem angegebenen Namen und den angegebenen Attributen erstellt. Die standardmäßige gewünschte Kapazität entspricht

der minimalen Größe. Daher startet diese Auto-Scaling-Gruppe zwei Instances – jeweils eine in jeder der angegebenen beiden Availability Zones.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -MinSize 2 -MaxSize 6 -AvailabilityZone @"us-west-2a", "us-west-2b")
```

- Einzelheiten zur API finden Sie unter [CreateAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def create_group(
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
    ) -> None:
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
```

```

        :param group_zones: The Availability Zones in which instances can be
        created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
        template.

        The launch template specifies the
        configuration of
        instances that are created by auto scaling
        activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling
        group.
    """
    try:
        self.autoscaling_client.create_auto_scaling_group(
            AutoScalingGroupName=group_name,
            AvailabilityZones=group_zones,
            LaunchTemplate={
                "LaunchTemplateName": launch_template_name,
                "Version": "$Default",
            },
            MinSize=min_size,
            MaxSize=max_size,
        )

        # Wait for the group to exist.
        waiter = self.autoscaling_client.get_waiter("group_exists")
        waiter.wait(AutoScalingGroupNames=[group_name])

        logger.info(f"Successfully created Auto Scaling group {group_name}.")

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(f"Failed to create Auto Scaling group {group_name}.")
        if error_code == "AlreadyExistsFault":
            logger.error(
                f"An Auto Scaling group with the name '{group_name}' already
                exists. "
                "Please use a different name or update the existing group.",
            )
        elif error_code == "LimitExceededFault":
            logger.error(
                "The request failed because you have reached the limit "

```

```

        "on the number of Auto Scaling groups or launch
        configurations. "
        "Consider deleting unused resources or request a limit
        increase. "
        "\nSee Auto Scaling Service Quota documentation here:"
        "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/
        ec2-auto-scaling-quotas.html"
    )
    logger.error(f"Full error:\n\t{err}")
    raise

```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error>
{
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}

```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
DATA lo_launch_template TYPE REF TO /aws1/cl_aslaunchtemplatespec.

" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_launch_template_name = 'my-launch-template'
" Example: iv_min_size = 1
" Example: iv_max_size = 3
" Example: iv_vpc_zone_identifier = 'subnet-12345,subnet-67890' (for VPC)

TRY.
  " Create launch template specification
  lo_launch_template = NEW /aws1/cl_aslaunchtemplatespec(
    iv_launchtemplatename = iv_launch_template_name
    iv_version = '$Default' ).

  " Create the Auto Scaling group
  " Use VPCZoneIdentifier for VPC subnets, or AvailabilityZones for EC2-
Classic
  IF iv_vpc_zone_identifier IS NOT INITIAL.
    " VPC-based deployment - use subnet IDs
    ao_asc->createautoscalinggroup(
      iv_autoscalinggroupname = iv_group_name
      iv_vpczoneidentifier = iv_vpc_zone_identifier
      io_launchtemplate = lo_launch_template
      iv_minsize = iv_min_size
      iv_maxsize = iv_max_size ).
  ELSE.
```

```
" EC2-Classic or default VPC - use availability zones
ao_asc->createautoscalinggroup(
  iv_autoscalinggroupname = iv_group_name
  it_availabilityzones = it_group_zones
  io_launchtemplate = lo_launch_template
  iv_minsize = iv_min_size
  iv_maxsize = iv_max_size ).
ENDIF.

" Wait for the group to be created (simplified - in production use proper
polling)
WAIT UP TO 10 SECONDS.

MESSAGE 'Auto Scaling group created successfully' TYPE 'I'.

CATCH /aws1/cx_ascalreadyexistsfault INTO DATA(lo_already_exists).
  RAISE EXCEPTION lo_already_exists.
CATCH /aws1/cx_asclimitexceededfault INTO DATA(lo_limit_exceeded).
  RAISE EXCEPTION lo_limit_exceeded.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
  RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **CreateLaunchConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLaunchConfiguration` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Startkonfiguration

In diesem Beispiel wird eine einfache Startkonfiguration erstellt.

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Erstellen einer Startkonfiguration](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So erstellen Sie eine Startkonfiguration mit einer Sicherheitsgruppe, einem key pair und einem Bootstrapping-Skript

In diesem Beispiel wird eine Startkonfiguration mit einer Sicherheitsgruppe, einem key pair und einem Bootstrapping-Skript erstellt, die in den Benutzerdaten enthalten sind.

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --security-groups sg-eb2af88example \  
  --key-name my-key-pair \  
  --user-data file://myuserdata.txt
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Erstellen einer Startkonfiguration](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So erstellen Sie eine Startkonfiguration mit einer IAM-Rolle

In diesem Beispiel wird eine Startkonfiguration mit dem Instance-Profilnamen einer IAM-Rolle erstellt.

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --iam-instance-profile my-autoscaling-role
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [IAM-Rolle für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 4: So erstellen Sie eine Startkonfiguration mit aktivierter detaillierter Überwachung

In diesem Beispiel wird eine Startkonfiguration mit aktivierter detaillierter EC2-Überwachung erstellt, an die EC2-Metriken innerhalb von 1 Minute gesendet werden CloudWatch .

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --instance-monitoring Enabled=true
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Konfigurieren der Überwachung für Auto-Scaling-Instances](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 5: So erstellen Sie eine Startkonfiguration, die Spot Instances startet

In diesem Beispiel wird eine Startkonfiguration erstellt, die Spot Instances als einzige Kaufoption verwendet.

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --spot-price "0.50"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Anfordern von Spot Instances](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 6: So erstellen Sie eine Startkonfiguration unter Verwendung einer EC2-Instance

In diesem Beispiel wird eine Startkonfiguration erstellt, die auf den Attributen einer vorhandenen Instance basiert. Es überschreibt die Tenancy für die Platzierung und ob eine öffentliche IP-Adresse festgelegt wird, indem die Optionen `--placement-tenancy` und `--no-associate-public-ip-address` einbezogen werden.

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc-from-instance \  
  --instance-id i-0123a456700123456 \  
  --instance-type m5.large \  
  --no-associate-public-ip-address \  
  --placement-tenancy dedicated
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Erstellen einer Startkonfiguration mithilfe einer EC2-Instance](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 7: So erstellen Sie eine Startkonfiguration mit einer Blockgerät-Zuweisung für ein Amazon EBS-Volume

In diesem Beispiel wird eine Startkonfiguration mit einer Blockgerät-Zuweisung für ein Amazon-EBSgp3-Volume mit dem Gerätenamen /dev/sdh und einer Volume-Größe von 20 erstellt.

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --block-device-mappings '[{"DeviceName":"/dev/sdh", "Ebs":  
  {"VolumeSize":20, "VolumeType":"gp3"}}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [EBS](#) in der API-Referenz für Amazon EC2 Auto Scaling.

Informationen zur Syntax für das Angeben von Parameterwerten im JSON-Format finden Sie unter [Verwenden von Anführungszeichen mit Zeichenfolgen in der AWS CLI im Benutzerhandbuch](#) für die AWS Befehlszeilenschnittstelle.

Beispiel 8: So erstellen Sie eine Startkonfiguration mit einer Blockgerät-Zuweisung für ein Instance-Speicher-Volume

In diesem Beispiel wird eine Startkonfiguration mit ephemeral1 als Instance-Speicher-Volume mit dem Gerätenamen /dev/sdc erstellt.

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --block-device-mappings '[{"DeviceName":"/dev/  
sdc", "VirtualName": "ephemeral1"}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie [BlockDeviceMapping](#) in der Amazon EC2 Auto Scaling API-Referenz.

Informationen zur Syntax für das Angeben von Parameterwerten im JSON-Format finden Sie unter [Verwenden von Anführungszeichen mit Zeichenfolgen in der AWS CLI im Benutzerhandbuch](#) für die AWS Befehlszeilenschnittstelle.

Beispiel 9: So erstellen Sie eine Startkonfiguration und verhindern, dass ein Blockgerät beim Start eine Verbindung herstellt

In diesem Beispiel wird eine Startkonfiguration erstellt, die das Blockgerät unterdrückt, das in der Blockgerät-Zuweisung des AMI angegeben ist (z. B. /dev/sdf).

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --block-device-mappings '[{"DeviceName":"/dev/sdf", "NoDevice": ""}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie [BlockDeviceMapping](#) in der Amazon EC2 Auto Scaling API-Referenz.

Informationen zur Syntax für das Angeben von Parameterwerten im JSON-Format finden Sie unter [Verwenden von Anführungszeichen mit Zeichenfolgen in der AWS CLI im Benutzerhandbuch](#) für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie unter [CreateLaunchConfiguration](#) Befehlsreferenz.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Startkonfiguration mit dem Namen „my-lc“ erstellt. Die von Auto-Scaling-Gruppen mit dieser Startkonfiguration gestarteten EC2-Instances verwenden den angegebenen Instance-Typ, das angegebene AMI, die Sicherheitsgruppe und die IAM-Rolle.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
  "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
  IamInstanceProfile "myIamRole"
```

- Einzelheiten zur API finden Sie unter [CreateLaunchConfiguration AWS -Tools für PowerShellCmdlet-Referenz \(V4\)](#).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Startkonfiguration mit dem Namen „my-lc“ erstellt. Die von Auto-Scaling-Gruppen mit dieser Startkonfiguration gestarteten EC2-Instances verwenden den angegebenen Instance-Typ, das angegebene AMI, die Sicherheitsgruppe und die IAM-Rolle.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
  "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
  IamInstanceProfile "myIamRole"
```

- Einzelheiten zur API finden Sie unter [CreateLaunchConfiguration AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **CreateOrUpdateTags** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateOrUpdateTags` verwendet wird.

CLI

AWS CLI

So erstellen oder ändern Sie Tags für eine vorhandene Auto-Scaling-Gruppe

In diesem Beispiel werden der angegebenen Auto-Scaling-Gruppe zwei Tags hinzugefügt.

```
aws autoscaling create-or-update-tags \  
  --tags ResourceId=my-asg,ResourceType=auto-scaling-  
group,Key=Role,Value=WebServer,PropagateAtLaunch=true ResourceId=my-  
asg,ResourceType=auto-scaling-  
group,Key=Dept,Value=Research,PropagateAtLaunch=true
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Markieren von Auto Scaling-Gruppen und Instances](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [CreateOrUpdateTags](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebenen Auto-Scaling-Gruppe ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist 'myTagValue'. Auto Scaling leitet dieses Tag an die nachfolgenden EC2-Instances weiter, die von der Auto-Scaling-Gruppe gestartet wurden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"
```

```
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
$tag.PropagateAtLaunch = $true  
Set-ASTag -Tag $tag
```

- Einzelheiten zur API finden Sie unter [CreateOrUpdateTags AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen Auto-Scaling-Gruppe ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist "myTagValue. Auto Scaling leitet dieses Tag an die nachfolgenden EC2-Instances weiter, die von der Auto-Scaling-Gruppe gestartet wurden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
$tag.PropagateAtLaunch = $true  
Set-ASTag -Tag $tag
```

- Einzelheiten zur API finden Sie unter [CreateOrUpdateTags AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **DeleteAutoScalingGroup** mit einem AWS SDK oder CLI


Die folgenden Code-Beispiele zeigen, wie `DeleteAutoScalingGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

SDK für .NET

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Aktualisieren Sie die Mindestgröße einer Auto-Scaling-Gruppe auf Null, beenden Sie alle Instances in der Gruppe und löschen Sie die Gruppe.

```
/// <summary>
/// Try to terminate an instance by its Id.
/// </summary>
/// <param name="instanceId">The Id of the instance to terminate.</param>
/// <returns>Async task.</returns>
public async Task TryTerminateInstanceById(string instanceId)
{
    var stopping = false;
    Console.WriteLine($"Stopping {instanceId}...");
    while (!stopping)
    {
        try
        {
            await
            _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                new TerminateInstanceInAutoScalingGroupRequest()
                {
                    InstanceId = instanceId,
                    ShouldDecrementDesiredCapacity = false
                });
            stopping = true;
        }
    }
}
```

```
        catch (ScalingActivityInProgressException)
        {
            Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
/// waits and retries until the group is successfully deleted.
/// </summary>
/// <param name="groupName">The name of the group to try to delete.</param>
/// <returns>Async task.</returns>
public async Task TryDeleteGroupByName(string groupName)
{
    var stopped = false;
    while (!stopped)
    {
        try
        {
            await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                new DeleteAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName
                });
            stopped = true;
        }
        catch (Exception e)
            when ((e is ScalingActivityInProgressException)
                || (e is Amazon.AutoScaling.Model.ResourceInUseException))
        {
            Console.WriteLine($"Some instances are still running.
Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Terminate instances and delete the Auto Scaling group by name.
/// </summary>
```

```

    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
        new DescribeAutoScalingGroupsRequest()
        {
            AutoScalingGroupNames = new List<string>() { groupName }
        });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
}

```

```

    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(

```

```
    string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
        _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
}
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
```

```
Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);

Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
    autoScalingClient.DeleteAutoScalingGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
               << outcome.GetError().GetMessage()
               << std::endl;
    result = false;
}
}
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So löschen Sie die angegebene Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht.

```
aws autoscaling delete-auto-scaling-group \
  --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Löschen der Auto-Scaling-Infrastruktur](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So erzwingen Sie das Löschen der angegebenen Auto-Scaling-Gruppe

Löschen Sie die Auto-Scaling-Gruppe, ohne darauf zu warten, dass die Instances in der Gruppe beendet werden, indem Sie die `--force-delete`-Option verwenden.

```
aws autoscaling delete-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --force-delete
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Löschen der Auto-Scaling-Infrastruktur](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;  
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;  
import  
  software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;  
  
/**  
 * Before running this SDK for Java (v2) code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-  
started.html  
 */  
public class DeleteAutoScalingGroup {  
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <groupName>

    Where:
        groupName - The name of the Auto Scaling group.
""";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String groupName = args[0];
AutoScalingClient autoScalingClient = AutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

deleteAutoScalingGroup(autoScalingClient, groupName);
autoScalingClient.close();
}

public static void deleteAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function deleteAutoScalingGroup($autoScalingGroupName)
{
    return $this->autoScalingClient->deleteAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'ForceDelete' => true,
    ]);
}
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht, sofern keine Instances ausgeführt werden. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
Target "my-asg".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Beispiel 3: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht und alle ausgeführten Instances, die sie enthält, werden beendet.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Einzelheiten zur API finden Sie unter [DeleteAutoScalingGroup AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht, sofern keine Instances ausgeführt werden. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
Target "my-asg".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Beispiel 3: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht und alle ausgeführten Instances, die sie enthält, werden beendet.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Einzelheiten zur API finden Sie unter [DeleteAutoScalingGroup AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Aktualisieren Sie die Mindestgröße einer Auto-Scaling-Gruppe auf Null, beenden Sie alle Instances in der Gruppe und löschen Sie die Gruppe.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
```

```

self.ami_param = ami_param
self.autoscaling_client = autoscaling_client
self.ec2_client = ec2_client
self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def delete_autoscaling_group(self, group_name: str) -> None:
    """
    Terminates all instances in the group, then deletes the EC2 Auto Scaling
    group.

    :param group_name: The name of the group to delete.
    """
    try:
        response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[group_name]
        )
        groups = response.get("AutoScalingGroups", [])
        if len(groups) > 0:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, MinSize=0
            )
            instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
            for inst_id in instance_ids:
                self.terminate_instance(inst_id)

```

```
# Wait for all instances to be terminated
if instance_ids:
    waiter = self.ec2_client.get_waiter("instance_terminated")
    log.info("Waiting for all instances to be terminated...")
    waiter.wait(InstanceIds=instance_ids)
    log.info("All instances have been terminated.")
else:
    log.info(f"No groups found named '{group_name}'! Nothing to do.")
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
    if error_code == "ScalingActivityInProgressFault":
        log.error(
            "Scaling activity is currently in progress. "
            "Wait for the scaling activity to complete before attempting
to delete the group again."
        )
    elif error_code == "ResourceContentionFault":
        log.error(
            "The request failed due to a resource contention issue. "
            "Ensure that no conflicting operations are being performed on
the group."
        )
    log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(),
Error> {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
}
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
    ao_asc->deleteautoscalinggroup(
        iv_autoscalinggroupname = iv_group_name ).

    " Wait for the group to be deleted (simplified - in production use proper
    polling)
    WAIT UP TO 10 SECONDS.

    MESSAGE 'Auto Scaling group deleted successfully' TYPE 'I'.

CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
```

```
RAISE EXCEPTION lo_activity_in_progress.  
CATCH /aws1/cx_ascresourceinusefault INTO DATA(lo_resource_in_use).  
RAISE EXCEPTION lo_resource_in_use.  
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).  
RAISE EXCEPTION lo_generic_exception.  
ENDTRY.
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteLaunchConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteLaunchConfiguration verwendet wird.

CLI

AWS CLI

So löschen Sie eine Startkonfiguration

In diesem Beispiel wird die angegebene Startkonfiguration gelöscht.

```
aws autoscaling delete-launch-configuration \  
  --launch-configuration-name my-launch-config
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Löschen der Auto-Scaling-Infrastruktur](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DeleteLaunchConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Startkonfiguration gelöscht, sofern sie nicht einer Auto-Scaling-Gruppe zugewiesen ist. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
on Target "my-lc".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Einzelheiten zur API finden Sie unter [DeleteLaunchConfiguration AWS -Tools für PowerShellCmdlet-Referenz \(V4\)](#).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Startkonfiguration gelöscht, sofern sie nicht einer Auto-Scaling-Gruppe zugewiesen ist. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
on Target "my-lc".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Einzelheiten zur API finden Sie unter [DeleteLaunchConfiguration AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteLifecycleHook** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteLifecycleHook verwendet wird.

CLI

AWS CLI

So löschen Sie einen Lebenszyklus-Hook

In diesem Beispiel wird der angegebene Lebenszyklus-Hook gelöscht.

```
aws autoscaling delete-lifecycle-hook \
  --lifecycle-hook-name my-lifecycle-hook \
  --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeleteLifecycleHook](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target  
"myLifecycleHook".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -Force
```

- Einzelheiten zur API finden Sie unter [DeleteLifecycleHook AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target  
"myLifecycleHook".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -Force
```

- Einzelheiten zur API finden Sie unter [DeleteLifecycleHook AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteNotificationConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteNotificationConfiguration verwendet wird.

CLI

AWS CLI

So löschen Sie eine Auto-Scaling-Benachrichtigung

In diesem Beispiel wird die angegebene Benachrichtigung aus der spezifizierten Auto-Scaling-Gruppe gelöscht.

```
aws autoscaling delete-notification-configuration \  
  --auto-scaling-group-name my-asg \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Löschen der Benachrichtigungskonfiguration](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DeleteNotificationConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Benachrichtigungsaktion gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN  
"arn:aws:sns:us-west-2:123456789012:my-topic"
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ASNotificationConfiguration  
(DeleteNotificationConfiguration)" on Target  
"arn:aws:sns:us-west-2:123456789012:my-topic".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN  
"arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Einzelheiten zur API finden Sie unter [DeleteNotificationConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Benachrichtigungsaktion gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN  
"arn:aws:sns:us-west-2:123456789012:my-topic"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
(DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
"arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Einzelheiten zur API finden Sie unter [DeleteNotificationConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeletePolicy** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeletePolicy verwendet wird.

CLI

AWS CLI

So löschen Sie eine Skalierungsrichtlinie

In diesem Beispiel wird die angegebene Scaling-Richtlinie gelöscht.

```
aws autoscaling delete-policy \
  --auto-scaling-group-name my-asg \
  --policy-name alb1000-target-tracking-scaling-policy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeletePolicy](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
"myScaleInPolicy".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Einzelheiten zur API finden Sie unter [DeletePolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target  
"myScaleInPolicy".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Einzelheiten zur API finden Sie unter [DeletePolicy AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteScheduledAction** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteScheduledAction verwendet wird.

CLI

AWS CLI

So löschen Sie eine geplante Aktion aus einer Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene geplante Aktion aus der spezifizierten Auto-Scaling-Gruppe gelöscht.

```
aws autoscaling delete-scheduled-action \  
  --auto-scaling-group-name my-asg \  
  --scheduled-action-name my-scheduled-action
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeleteScheduledAction](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene geplante Aktion für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction"
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target  
"myScheduledAction".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction" -Force
```

- Einzelheiten zur API finden Sie unter [DeleteScheduledAction AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene geplante Aktion für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction"
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target  
"myScheduledAction".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction" -Force
```

- Einzelheiten zur API finden Sie unter [DeleteScheduledAction AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteTags** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteTags verwendet wird.

CLI

AWS CLI

So löschen Sie ein Tag aus einer Auto-Scaling-Gruppe

In diesem Beispiel wird das angegebene Tag aus der angegebenen Auto-Scaling-Gruppe gelöscht.

```
aws autoscaling delete-tags \  
  --tags ResourceId=my-asg,ResourceType=auto-scaling-  
group,Key=Dept,Value=Research
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Markieren von Auto Scaling-Gruppen und Instances](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DeleteTags](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Tag der spezifizierten Auto-Scaling-Gruppe entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag" } )
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-ASTag (DeleteTags)" on target  
"Amazon.AutoScaling.Model.Tag".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag" } ) -Force
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
Remove-ASTag -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter [DeleteTags AWS -Tools für PowerShell Cmdlet-Referenz \(V4\)](#).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag der spezifizierten Auto-Scaling-Gruppe entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag" } )
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-ASTag (DeleteTags)" on target  
"Amazon.AutoScaling.Model.Tag".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag" } ) -Force
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
Remove-ASTag -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter [DeleteTags AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeAccountLimits** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAccountLimits` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Limits Ihres Kontos bei Amazon EC2 Auto Scaling

In diesem Beispiel werden die Amazon EC2 Auto Scaling Scaling-Limits für Ihr AWS Konto beschrieben.

```
aws autoscaling describe-account-limits
```

Ausgabe:

```
{
  "NumberOfLaunchConfigurations": 5,
  "MaxNumberOfLaunchConfigurations": 100,
  "NumberOfAutoScalingGroups": 3,
  "MaxNumberOfAutoScalingGroups": 20
}
```

Weitere Informationen finden Sie unter [Servicekontingente für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeAccountLimits](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Auto Scaling Scaling-Ressourcenlimits für Ihr AWS Konto beschrieben.

```
Get-ASAccountLimit
```

Ausgabe:

```
MaxNumberOfAutoScalingGroups : 20
```

```
MaxNumberOfLaunchConfigurations : 100
```

- Einzelheiten zur API finden Sie unter [DescribeAccountLimits AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Auto Scaling Scaling-Ressourcenlimits für Ihr AWS Konto beschrieben.

```
Get-ASAccountLimit
```

Ausgabe:

```
MaxNumberOfAutoScalingGroups      : 20  
MaxNumberOfLaunchConfigurations   : 100
```

- Einzelheiten zur API finden Sie unter [DescribeAccountLimits AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeAdjustmentTypes** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAdjustmentTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Typen der Skalierungsanpassung

In diesem Beispiel werden die verfügbaren Anpassungstypen beschrieben.

```
aws autoscaling describe-adjustment-types
```

Ausgabe:

```
{
```

```
"AdjustmentTypes": [  
  {  
    "AdjustmentType": "ChangeInCapacity"  
  },  
  {  
    "AdjustmentType": "ExactCapacity"  
  },  
  {  
    "AdjustmentType": "PercentChangeInCapacity"  
  }  
]
```

Weitere Informationen finden Sie unter [Skalierungsanpassungstypen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeAdjustmentTypes](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Anpassungstypen beschrieben, die von Auto Scaling unterstützt werden.

```
Get-ASAdjustmentType
```

Ausgabe:

```
Type  
----  
ChangeInCapacity  
ExactCapacity  
PercentChangeInCapacity
```

- Einzelheiten zur API finden Sie unter [DescribeAdjustmentTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Anpassungstypen beschrieben, die von Auto Scaling unterstützt werden.

```
Get-ASAdjustmentType
```

Ausgabe:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Einzelheiten zur API finden Sie unter [DescribeAdjustmentTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **DescribeAutoScalingGroups** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAutoScalingGroups` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
///  
/// <summary>
```

```
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
    client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                << outcome.GetError().GetMessage()
                << std::endl;
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie die angegebene Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-auto-scaling-groups \  
  --auto-scaling-group-names my-asg
```

Ausgabe:

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-  
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-  
a11a-7b0acd480f03:autoScalingGroupName/my-asg",  
      "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1",  
        "LaunchTemplateId": "lt-1234567890abcde12"  
      },  
      "MinSize": 0,  
      "MaxSize": 1,  
      "DesiredCapacity": 1,  
      "DefaultCooldown": 300,  
      "AvailabilityZones": [  
        "us-west-2a",  
        "us-west-2b",  
        "us-west-2c"  
      ],  
      "LoadBalancerNames": [],  
      "TargetGroupARNs": [],  
      "HealthCheckType": "EC2",  
      "HealthCheckGracePeriod": 0,  
      "Instances": [  
        {  
          "InstanceId": "i-06905f55584de02da",  
          "InstanceType": "t2.micro",  
          "AvailabilityZone": "us-west-2a",
```

```

        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "ProtectedFromScaleIn": false,
        "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1",
            "LaunchTemplateId": "lt-1234567890abcde12"
        }
    },
    "CreatedTime": "2023-10-28T02:39:22.152Z",
    "SuspendedProcesses": [],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-
c934b782",
    "EnabledMetrics": [],
    "Tags": [],
    "TerminationPolicies": [
        "Default"
    ],
    "NewInstancesProtectedFromScaleIn": false,
    "ServiceLinkedRoleARN": "arn",
    "TrafficSources": []
}
]
}

```

Beispiel 2: So beschreiben Sie die ersten 100 angegebenen Auto-Scaling-Gruppen

In diesem Beispiel werden die angegebenen Auto-Scaling-Gruppen beschrieben. Sie können bis zu 100 Gruppennamen angeben.

```

aws autoscaling describe-auto-scaling-groups \
  --max-items 100 \
  --auto-scaling-group-names "group1" "group2" "group3" "group4"

```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 3: So beschreiben Sie eine Auto-Scaling-Gruppe in der angegebenen Region

In diesem Beispiel werden die Auto-Scaling-Gruppen in der angegebenen Region beschrieben, wobei maximal 75 Gruppen berücksichtigt werden.

```

aws autoscaling describe-auto-scaling-groups \

```

```
--max-items 75 \  
--region us-east-1
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 4: So beschreiben Sie die angegebene Anzahl von Auto-Scaling-Gruppen

Um eine bestimmte Anzahl von Auto-Scaling-Gruppen zurückzugeben, verwenden Sie die `--max-items`-Option.

```
aws autoscaling describe-auto-scaling-groups \  
--max-items 1
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere Gruppen. Um die zusätzlichen Gruppen abzurufen, verwenden Sie den Wert dieses Felds mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt.

```
aws autoscaling describe-auto-scaling-groups \  
--starting-token Z3M3LMPEXAMPLE
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 5: So beschreiben Sie Auto-Scaling-Gruppen, die Startkonfigurationen verwenden

Dieses Beispiel verwendet die `--query`-Option, um Auto-Scaling-Gruppen zu beschreiben, die Startkonfigurationen verwenden.

```
aws autoscaling describe-auto-scaling-groups \  
--query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

Ausgabe:

```
[  
  {  
    "AutoScalingGroupName": "my-asg",  
    "AutoScalingGroupARN": "arn:aws:autoscaling:us-  
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-  
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
```

```

    "LaunchConfigurationName": "my-lc",
    "MinSize": 0,
    "MaxSize": 1,
    "DesiredCapacity": 1,
    "DefaultCooldown": 300,
    "AvailabilityZones": [
      "us-west-2a",
      "us-west-2b",
      "us-west-2c"
    ],
    "LoadBalancerNames": [],
    "TargetGroupARNs": [],
    "HealthCheckType": "EC2",
    "HealthCheckGracePeriod": 0,
    "Instances": [
      {
        "InstanceId": "i-088c57934a6449037",
        "InstanceType": "t2.micro",
        "AvailabilityZone": "us-west-2c",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc",
        "ProtectedFromScaleIn": false
      }
    ],
    "CreatedTime": "2023-10-28T02:39:22.152Z",
    "SuspendedProcesses": [],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "EnabledMetrics": [],
    "Tags": [],
    "TerminationPolicies": [
      "Default"
    ],
    "NewInstancesProtectedFromScaleIn": false,
    "ServiceLinkedRoleARN": "arn",
    "TrafficSources": []
  }
]

```

Weitere Informationen finden Sie unter [AWS CLI-Ausgabe filtern](#) im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) unter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName>

            Where:
                groupName - The name of the Auto Scaling group.
            """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String groupName = args[0];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String instanceId = getAutoScaling(autoScalingClient, groupName);
    System.out.println(instanceId);
    autoScalingClient.close();
}

public static String getAutoScaling(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response = autoScalingClient
            .describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " +
group.autoScalingGroupARN());

            List<Instance> instances = group.instances();
            for (Instance instance : instances) {
                instanceId = instance.instanceId();
            }
        }
        return instanceId;
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}

```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
            response.autoScalingGroups?.forEach { group ->
                println("The group name is ${group.autoScalingGroupName}")
                println("The group ARN is ${group.autoScalingGroupArn}")
                group.instances?.forEach { instance ->
                    println("The instance id is ${instance.instanceId}")
                    println("The lifecycle state is " + instance.lifecycleState)
                }
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function describeAutoScalingGroups($autoScalingGroupNames)
{
    return $this->autoScalingClient->describeAutoScalingGroups([
        'AutoScalingGroupNames' => $autoScalingGroupNames
    ]);
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Auto-Scaling-Gruppen aufgeführt.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Ausgabe:

```
AutoScalingGroupName
-----
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Ausgabe:

```
AutoScalingGroupARN      : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                           f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName     : my-asg-1
AvailabilityZones        : {us-west-2b, us-west-2a}
CreatedTime              : 3/1/2015 9:05:31 AM
DefaultCooldown          : 300
DesiredCapacity          : 2
EnabledMetrics           : {}
HealthCheckGracePeriod   : 300
HealthCheckType          : EC2
Instances                : {my-1c}
LaunchConfigurationName  : my-1c
LoadBalancerNames       : {}
MaxSize                  : 0
MinSize                  : 0
PlacementGroup           :
Status                   :
SuspendedProcesses       : {}
Tags                    : {}
TerminationPolicies      : {Default}
VPCZoneIdentifier        : subnet-e4f33493,subnet-5264e837
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @"my-asg-1", "my-asg-2"
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup
```

Beispiel 6: Dieses LaunchTemplate Beispiel beschreibt die angegebene Auto Scaling Scaling-Gruppe. In diesem Beispiel wird davon ausgegangen, dass die Option „Instance-Kaufoptionen“ auf „Startvorlage beachten“ eingestellt ist. Falls diese Option auf „Kaufoptionen und Instanztypen kombinieren“ gesetzt ist, LaunchTemplate kann mit "darauf zugegriffen werdenMixedInstancesPolicy. LaunchTemplate,-Eigenschaft.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Ausgabe:

```
LaunchTemplateId      LaunchTemplateName    Version
-----
lt-06095fd619cb40371 test-launch-template  $Default
```

- Einzelheiten zur API finden Sie unter [DescribeAutoScalingGroups AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Auto-Scaling-Gruppen aufgeführt.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Ausgabe:

```
AutoScalingGroupName
-----
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Ausgabe:

```

AutoScalingGroupARN      : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                           f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName     : my-asg-1
AvailabilityZones        : {us-west-2b, us-west-2a}
CreatedTime              : 3/1/2015 9:05:31 AM
DefaultCooldown          : 300
DesiredCapacity          : 2
EnabledMetrics           : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances                : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                  : 0
MinSize                  : 0
PlacementGroup          :
Status                   :
SuspendedProcesses      : {}
Tags                     : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier        : subnet-e4f33493,subnet-5264e837

```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @"("my-asg-1", "my-asg-2")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup
```

Beispiel 6: Dieses LaunchTemplate Beispiel beschreibt die angegebene Auto Scaling Scaling-Gruppe. In diesem Beispiel wird davon ausgegangen, dass die Option „Instance-

Kaufoptionen“ auf „Startvorlage beachten“ eingestellt ist. Falls diese Option auf „Kaufoptionen und Instanztypen kombinieren“ gesetzt ist, LaunchTemplate kann mit "darauf zugegriffen werdenMixedInstancesPolicy. LaunchTemplate,-Eigenschaft.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Ausgabe:

LaunchTemplateId	LaunchTemplateName	Version
-----	-----	-----
lt-06095fd619cb40371	test-launch-template	\$Default

- Einzelheiten zur API finden Sie unter [DescribeAutoScalingGroups AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
        """
        Gets information about an Auto Scaling group.
```

```

        :param group_name: The name of the group to look up.
        :return: A dictionary with information about the group if found,
otherwise None.
        :raises ClientError: If there is an error describing the Auto Scaling
group.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_groups"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupNames=[group_name])
            groups = []
            for response in response_iterator:
                groups.extend(response.get("AutoScalingGroups", []))

            logger.info(
                f"Successfully retrieved information for Auto Scaling group
{group_name}."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to describe Auto Scaling group {group_name}.")
            if error_code == "ResourceContentionFault":
                logger.error(
                    "There is a conflict with another operation that is modifying
the "
                    f"Auto Scaling group '{group_name}' Please try again later."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
        else:
            return groups[0] if len(groups) > 0 else None

```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name: {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn: {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }


    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

 Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

DATA lt_group_names TYPE /aws1/
cl_ascautoscgroupnames_w=>tt_autoscalinggroupnames.
DATA lo_group_name TYPE REF TO /aws1/cl_ascautoscgroupnames_w.

" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
  " Build group names parameter
  CREATE OBJECT lo_group_name
    EXPORTING
      iv_value = iv_group_name.
  APPEND lo_group_name TO lt_group_names.

  " Describe the Auto Scaling group
  DATA(lo_output) = ao_asc->describeautoscalinggroups(
    it_autoscalinggroupnames = lt_group_names ).

  " Return the first (and only) group in the result
  DATA(lt_groups) = lo_output->get_autoscalinggroups( ).
  IF lines( lt_groups ) > 0.
    READ TABLE lt_groups INDEX 1 INTO DATA(lo_group).
    oo_output = lo_group.
  ENDIF.

  MESSAGE 'Auto Scaling group information retrieved successfully' TYPE 'I'.

  CATCH /aws1/cx_asresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.

```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **DescribeAutoScalingInstances** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAutoScalingInstances` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Get data about the instances in an Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
public async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    string groupName)
{
    var groups = await DescribeAutoScalingGroupsAsync(groupName);
    var instanceIds = new List<string>();
    var instanceDetails = new List<AutoScalingInstanceDetails>();
```

```
    if (groups != null)
    {
        groups.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };


        var response =
            await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                scalingGroupsRequest);
        if (response.AutoScalingInstances != null)
        {
            instanceDetails = response.AutoScalingInstances;
        }
    }

    return instanceDetails;
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
request.SetInstanceIds(instanceIDs);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
    client.DescribeAutoScalingInstances(request);

if (outcome.IsSuccess()) {
    const
    Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
    &instancesDetails =
        outcome.GetResult().GetAutoScalingInstances();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                << outcome.GetError().GetMessage()
                << std::endl;
    return false;
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine oder mehrere Instances

In diesem Beispiel wird die angegebene Instance beschrieben.

```
aws autoscaling describe-auto-scaling-instances \  
  --instance-ids i-06905f55584de02da
```

Ausgabe:

```
{  
  "AutoScalingInstances": [  
    {  
      "InstanceId": "i-06905f55584de02da",  
      "InstanceType": "t2.micro",  
      "AutoScalingGroupName": "my-asg",  
      "AvailabilityZone": "us-west-2b",  
      "LifecycleState": "InService",  
      "HealthStatus": "HEALTHY",  
      "ProtectedFromScaleIn": false,  
      "LaunchTemplate": {  
        "LaunchTemplateId": "lt-1234567890abcde12",  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1"  
      }  
    }  
  ]  
}
```

Beispiel 2: So beschreiben Sie eine oder mehrere Instances

Dieses Beispiel verwendet die `--max-items`-Option, um anzugeben, wie viele Instances mit diesem Aufruf zurückgegeben werden sollen.

```
aws autoscaling describe-auto-scaling-instances \  
  --max-items 1
```

Wenn die Ausgabe ein NextToken-Feld enthält, gibt es weitere Instances. Um die zusätzlichen Instances abzurufen, verwenden Sie den Wert dieses Felds mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt.

```
aws autoscaling describe-auto-scaling-instances \  
  --starting-token Z3M3LMPEXAMPLE
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 3: So beschreiben Sie Instances, die Startkonfigurationen verwenden

In diesem Beispiel wird die `--query`-Option verwendet, um Instances zu beschreiben, die Startkonfigurationen verwenden.

```
aws autoscaling describe-auto-scaling-instances \  
  --query 'AutoScalingInstances[?LaunchConfigurationName!=`null`]'
```

Ausgabe:

```
[  
  {  
    "InstanceId": "i-088c57934a6449037",  
    "InstanceType": "t2.micro",  
    "AutoScalingGroupName": "my-asg",  
    "AvailabilityZone": "us-west-2c",  
    "LifecycleState": "InService",  
    "HealthStatus": "HEALTHY",  
    "LaunchConfigurationName": "my-lc",  
    "ProtectedFromScaleIn": false  
  }  
]
```

Weitere Informationen finden Sie unter [AWS CLI-Ausgabe filtern](#) im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) unter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function describeAutoScalingInstances($instanceIds)
```

```
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Ihre Auto Scaling Scaling-Instances aufgeführt. IDs

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Ausgabe:

```
InstanceId
-----
i-12345678
i-87654321
i-abcd1234
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Instance beschrieben.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Ausgabe:

```
AutoScalingGroupName    : my-asg
AvailabilityZone         : us-west-2b
HealthStatus            : HEALTHY
InstanceId               : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance -InstanceId @"(i-12345678", "i-87654321")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-ASAutoScalingInstance
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance
```

- Einzelheiten zur API finden Sie unter [DescribeAutoScalingInstances AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Ihre Auto Scaling Scaling-Instances aufgeführt. IDs

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Ausgabe:

```
InstanceId
-----
i-12345678
i-87654321
i-abcd1234
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Instance beschrieben.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Ausgabe:

```
AutoScalingGroupName    : my-asg
AvailabilityZone         : us-west-2b
HealthStatus             : HEALTHY
```

```
InstanceId           : i-12345678
LaunchConfigurationName : my-lc
LifecycleState       : InService
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance -InstanceId @"(i-12345678", "i-87654321")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-ASAutoScalingInstance
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance
```

- Einzelheiten zur API finden Sie unter [DescribeAutoScalingInstances AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
```

```
"""
self.autoscaling_client = autoscaling_client

def describe_instances(self, instance_ids: List[str]) -> List[Dict[str,
Any]]:
    """
    Gets information about instances.

    :param instance_ids: A list of instance IDs to look up.
    :return: A list of dictionaries with information about each instance,
             or an empty list if none are found.
    :raises ClientError: If there is an error describing the instances.
    """
    try:
        paginator = self.autoscaling_client.get_paginator(
            "describe_auto_scaling_instances"
        )
        response_iterator = paginator.paginate(InstanceIds=instance_ids)

        instances = []
        for response in response_iterator:
            instances.extend(response.get("AutoScalingInstances", []))

        logger.info(f"Successfully described instances: {instance_ids}")

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(
            f"Couldn't describe instances {instance_ids}. Error code:
            {error_code}, Message: {err.response['Error']['Message']}"
        )
        raise
    else:
        return instances
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
    // The direct way to list instances is by using
    DescribeAutoScalingGroup's instances property. However, this returns a
    Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
    // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

    // Alternatively, and for the sake of example,
    DescribeAutoScalingInstances returns a list that can be filtered by the client.
    self.autoscaling
        .describe_auto_scaling_instances()
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await
        .map(|items| {
            items
                .into_iter()
                .filter(|i| {
                    i.auto_scaling_group_name.as_deref()
                        == Some(self.auto_scaling_group_name.as_str())
                })
                .map(|i| i.instance_id.unwrap_or_default())
                .filter(|id| !id.is_empty())
                .collect:::<Vec<String>>()
        })
        .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
" Example: it_instance_ids contains a list of instance IDs

TRY.
  DATA(lo_output) = ao_asc->describeautoscalinginstances(
    it_instanceids = it_instance_ids ).

  ot_output = lo_output->get_autoscalinginstances( ).

  MESSAGE 'Auto Scaling instances information retrieved successfully' TYPE
'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von `DescribeAutoScalingNotificationTypes` mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAutoScalingNotificationTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Benachrichtigungstypen

In diesem Beispiel werden die verfügbaren Benachrichtigungstypen beschrieben.

```
aws autoscaling describe-auto-scaling-notification-types
```

Ausgabe:

```
{
  "AutoScalingNotificationTypes": [
    "autoscaling:EC2_INSTANCE_LAUNCH",
    "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
    "autoscaling:EC2_INSTANCE_TERMINATE",
    "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
    "autoscaling:TEST_NOTIFICATION"
  ]
}
```

Weitere Informationen finden Sie unter [Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeAutoScalingNotificationTypes](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Benachrichtigungstypen aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASAutoScalingNotificationType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Einzelheiten zur API finden Sie unter [DescribeAutoScalingNotificationTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Benachrichtigungstypen aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASAutoScalingNotificationType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Einzelheiten zur API finden Sie unter [DescribeAutoScalingNotificationTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Verwenden Sie diesen Service mit einem SDK AWS](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von `DescribeLaunchConfigurations` mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLaunchConfigurations` verwendet wird.

CLI**AWS CLI**

Beispiel 1: So beschreiben Sie die angegebene Startkonfiguration

In diesem Beispiel wird die angegebene Startkonfiguration beschrieben.

```
aws autoscaling describe-launch-configurations \
  --launch-configuration-names my-launch-config
```

Ausgabe:

```
{
  "LaunchConfigurations": [
    {
      "LaunchConfigurationName": "my-launch-config",
      "LaunchConfigurationARN": "arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:98d3b196-4cf9-4e88-8ca1-8547c24ced8b:launchConfig
my-launch-config",
      "ImageId": "ami-0528a5175983e7f28",
      "KeyName": "my-key-pair-uswest2",
      "SecurityGroups": [
        "sg-05eaec502fcdadc2e"
      ],
      "ClassicLinkVPCSecurityGroups": [],
      "UserData": "",
      "InstanceType": "t2.micro",
      "KernelId": "",
      "RamdiskId": "",
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/xvda",
          "Ebs": {
            "SnapshotId": "snap-06c1606ba5ca274b1",
            "VolumeSize": 8,
            "VolumeType": "gp2",
            "DeleteOnTermination": true,
            "Encrypted": false
          }
        }
      ],
      "InstanceMonitoring": {
        "Enabled": true
      },
      "CreatedTime": "2020-10-28T02:39:22.321Z",
      "EbsOptimized": false,
      "AssociatePublicIpAddress": true,
      "MetadataOptions": {
```

```

        "HttpTokens": "required",
        "HttpPutResponseHopLimit": 1,
        "HttpEndpoint": "disabled"
    }
}
]
}

```

Beispiel 2: So beschreiben Sie eine bestimmte Anzahl von Startkonfigurationen

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von Startkonfigurationen zurückzugeben.

```

aws autoscaling describe-launch-configurations \
  --max-items 1

```

Wenn die Ausgabe ein `NextToken`-Feld enthält, stehen weitere Startkonfigurationen zur Verfügung. Um die zusätzlichen Startkonfigurationen abzurufen, verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt.

```

aws autoscaling describe-launch-configurations \
  --starting-token Z3M3LMPEXAMPLE

```

- Einzelheiten zur API finden Sie [DescribeLaunchConfigurations](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Startkonfigurationen aufgeführt.

```

Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName

```

Ausgabe:

```

LaunchConfigurationName
-----
my-lc-1
my-lc-2

```

```
my-lc-3
my-lc-4
my-lc-5
```

Beispiel 2: In diesem Beispiel wird die angegebene Startkonfiguration beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Ausgabe:

```
AssociatePublicIpAddress      : True
BlockDeviceMappings           : {/dev/xvda}
ClassicLinkVPCId              :
ClassicLinkVPCSecurityGroups  : {}
CreatedTime                   : 12/12/2014 3:22:08 PM
EbsOptimized                   : False
IamInstanceProfile            :
ImageId                       : ami-043a5034
InstanceMonitoring            : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                   : t2.micro
KernelId                      :
KeyName                       :
LaunchConfigurationARN        : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName       : my-lc-1
PlacementTenancy              :
RamdiskId                     :
SecurityGroups                 : {sg-67ef0308}
SpotPrice                     :
UserData                      :
```

Beispiel 3: In diesem Beispiel werden die beiden angegebenen Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Beispiel 4: In diesem Beispiel werden alle Ihre Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter [DescribeLaunchConfigurations AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Startkonfigurationen aufgeführt.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Ausgabe:

```
LaunchConfigurationName
-----
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Beispiel 2: In diesem Beispiel wird die angegebene Startkonfiguration beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Ausgabe:

```
AssociatePublicIpAddress      : True
BlockDeviceMappings           : {/dev/xvda}
ClassicLinkVPCId              :
ClassicLinkVPCSecurityGroups  : {}
CreatedTime                   : 12/12/2014 3:22:08 PM
EbsOptimized                  : False
IamInstanceProfile            :
ImageId                       : ami-043a5034
InstanceMonitoring            : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                  : t2.micro
KernelId                     :
KeyName                       :
LaunchConfigurationARN        : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
e6f68d7fafad:launchConfiguration/my-lc-1
LaunchConfigurationName       : my-lc-1
PlacementTenancy              :
```

```
RamdiskId           :  
SecurityGroups      : {sg-67ef0308}  
SpotPrice           :  
UserData            :
```

Beispiel 3: In diesem Beispiel werden die beiden angegebenen Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Beispiel 4: In diesem Beispiel werden alle Ihre Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter [DescribeLaunchConfigurations AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeLifecycleHookTypes** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLifecycleHookTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Typen von Lebenszyklus-Hooks

In diesem Beispiel werden die verfügbaren Typen von Lebenszyklus-Hooks beschrieben.

```
aws autoscaling describe-lifecycle-hook-types
```

Ausgabe:

```
{  
  "LifecycleHookTypes": [  
    ...  
  ]  
}
```

```
        "autoscaling:EC2_INSTANCE_LAUNCHING",  
        "autoscaling:EC2_INSTANCE_TERMINATING"  
    ]  
}
```

- Einzelheiten zur API finden Sie [DescribeLifecycleHookTypes](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die von Auto Scaling unterstützten Typen von Lebenszyklus-Hooks aufgeführt.

```
Get-ASLifecycleHookType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCHING  
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Einzelheiten zur API finden Sie unter [DescribeLifecycleHookTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die von Auto Scaling unterstützten Typen von Lebenszyklus-Hooks aufgeführt.

```
Get-ASLifecycleHookType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCHING  
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Einzelheiten zur API finden Sie unter [DescribeLifecycleHookTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Verwenden Sie diesen Service mit einem SDK AWS](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeLifecycleHooks** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLifecycleHooks` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre Lebenszyklus-Hooks

In diesem Beispiel werden die Lebenszyklus-Hooks für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-lifecycle-hooks \  
  --auto-scaling-group-name my-asg
```

Ausgabe:

```
{  
  "LifecycleHooks": [  
    {  
      "GlobalTimeout": 3000,  
      "HeartbeatTimeout": 30,  
      "AutoScalingGroupName": "my-asg",  
      "LifecycleHookName": "my-launch-hook",  
      "DefaultResult": "ABANDON",  
      "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING"  
    },  
    {  
      "GlobalTimeout": 6000,  
      "HeartbeatTimeout": 60,  
      "AutoScalingGroupName": "my-asg",  
      "LifecycleHookName": "my-termination-hook",  
      "DefaultResult": "CONTINUE",  
      "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING"  
    }  
  ]  
}
```

- Einzelheiten zur API finden Sie [DescribeLifecycleHooks](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook
```

Ausgabe:

```
AutoScalingGroupName : my-asg  
DefaultResult        : ABANDON  
GlobalTimeout        : 172800  
HeartbeatTimeout     : 3600  
LifecycleHookName    : myLifecycleHook  
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING  
NotificationMetadata :  
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic  
RoleARN              : arn:aws:iam::123456789012:role/my-iam-role
```

Beispiel 2: In diesem Beispiel werden alle Lebenszyklus-Hooks für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Beispiel 3: In diesem Beispiel werden alle Lebenszyklus-Hooks für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASLifecycleHook
```

- Einzelheiten zur API finden Sie unter [DescribeLifecycleHooks AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
myLifecycleHook
```

Ausgabe:

```
AutoScalingGroupName : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

Beispiel 2: In diesem Beispiel werden alle Lebenszyklus-Hooks für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Beispiel 3: In diesem Beispiel werden alle Lebenszyklus-Hooks für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASLifecycleHook
```

- Einzelheiten zur API finden Sie unter [DescribeLifecycleHooks AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeLoadBalancers** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLoadBalancers` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Classic Load Balancers für eine Auto-Scaling-Gruppe

In diesem Beispiel werden die Classic Load Balancers für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-load-balancers \  
  --auto-scaling-group-name my-asg
```

Ausgabe:

```
{  
  "LoadBalancers": [  
    {  
      "State": "Added",  
      "LoadBalancerName": "my-load-balancer"  
    }  
  ]  
}
```

- Einzelheiten zur API finden Sie [DescribeLoadBalancers](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Load Balancers für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Ausgabe:

LoadBalancerName	State
-----	-----
my-lb	Added

- Einzelheiten zur API finden Sie unter [DescribeLoadBalancers AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Load Balancers für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Ausgabe:

```
LoadBalancerName    State
-----
my-lb                Added
```

- Einzelheiten zur API finden Sie unter [DescribeLoadBalancers AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeMetricCollectionTypes** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMetricCollectionTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Arten der Metrikerfassung

In diesem Beispiel werden die verfügbaren Arten der Metrikerfassung beschrieben.

```
aws autoscaling describe-metric-collection-types
```

Ausgabe:

```
{
  "Metrics": [
    {
```

```
    "Metric": "GroupMinSize"
  },
  {
    "Metric": "GroupMaxSize"
  },
  {
    "Metric": "GroupDesiredCapacity"
  },
  {
    "Metric": "GroupInServiceInstances"
  },
  {
    "Metric": "GroupInServiceCapacity"
  },
  {
    "Metric": "GroupPendingInstances"
  },
  {
    "Metric": "GroupPendingCapacity"
  },
  {
    "Metric": "GroupTerminatingInstances"
  },
  {
    "Metric": "GroupTerminatingCapacity"
  },
  {
    "Metric": "GroupStandbyInstances"
  },
  {
    "Metric": "GroupStandbyCapacity"
  },
  {
    "Metric": "GroupTotalInstances"
  },
  {
    "Metric": "GroupTotalCapacity"
  }
],
"Granularities": [
  {
    "Granularity": "1Minute"
  }
]
```

```
}
```

Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen-Metriken](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeMetricCollectionTypes](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Arten der Metrikerfassung aufgeführt, die von Auto Scaling unterstützt werden.

```
(Get-ASMetricCollectionType).Metrics
```

Ausgabe:

```
Metric
-----
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Beispiel 2: In diesem Beispiel werden die entsprechenden Granularitäten aufgeführt.

```
(Get-ASMetricCollectionType).Granularities
```

Ausgabe:

```
Granularity
-----
1Minute
```

- Einzelheiten zur API finden Sie unter [DescribeMetricCollectionTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Arten der Metrikerfassung aufgeführt, die von Auto Scaling unterstützt werden.

```
(Get-ASMetricCollectionType).Metrics
```

Ausgabe:

```
Metric
-----
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Beispiel 2: In diesem Beispiel werden die entsprechenden Granularitäten aufgeführt.

```
(Get-ASMetricCollectionType).Granularities
```

Ausgabe:

```
Granularity
-----
1Minute
```

- Einzelheiten zur API finden Sie unter [DescribeMetricCollectionTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeNotificationConfigurations** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeNotificationConfigurations` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie die Benachrichtigungskonfigurationen einer angegebenen Gruppe

In diesem Beispiel werden die Benachrichtigungskonfigurationen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-notification-configurations \  
  --auto-scaling-group-name my-asg
```

Ausgabe:

```
{  
  "NotificationConfigurations": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "NotificationType": "autoscaling:TEST_NOTIFICATION",  
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"  
    },  
    {  
      "AutoScalingGroupName": "my-asg",  
      "NotificationType": "autoscaling:TEST_NOTIFICATION",  
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 1: So beschreiben Sie eine bestimmte Anzahl von Benachrichtigungskonfigurationen

Verwenden Sie den `max-items`-Parameter, um eine bestimmte Anzahl von Benachrichtigungskonfigurationen zurückzugeben.

```
aws autoscaling describe-notification-configurations \  
  --auto-scaling-group-name my-auto-scaling-group \  
  --max-items 1
```

Ausgabe:

```
{  
  "NotificationConfigurations": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "NotificationType": "autoscaling:TEST_NOTIFICATION",  
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"  
    },  
    {  
      "AutoScalingGroupName": "my-asg",  
      "NotificationType": "autoscaling:TEST_NOTIFICATION",  
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"  
    }  
  ]  
}
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere Benachrichtigungskonfigurationen. Verwenden Sie den Wert dieses Feldes mit dem `starting-token`-Parameter in einem nachfolgenden Aufruf wie folgt, um die zusätzlichen Benachrichtigungskonfigurationen zu erhalten.

```
aws autoscaling describe-notification-configurations \  
  --auto-scaling-group-name my-asg \  
  --starting-token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter [Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeNotificationConfigurations](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt die Benachrichtigungsaktionen, die der angegebenen Auto Scaling-Gruppe zugewiesen sind.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Ausgabe:

```
AutoScalingGroupName : my-asg
NotificationType      : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType      : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic
```

Beispiel 2: In diesem Beispiel werden die Benachrichtigungsaktionen beschrieben, die allen Ihren Auto-Scaling-Gruppen zugewiesen sind.

```
Get-ASNotificationConfiguration
```

- Einzelheiten zur API finden Sie unter [DescribeNotificationConfigurations AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Benachrichtigungsaktionen, die der angegebenen Auto Scaling-Gruppe zugewiesen sind.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Ausgabe:

```
AutoScalingGroupName : my-asg
NotificationType      : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType      : auto-scaling:EC2_INSTANCE_TERMINATE
```

```
TopicARN : arn:aws:sns:us-west-2:123456789012:my-topic
```

Beispiel 2: In diesem Beispiel werden die Benachrichtigungsaktionen beschrieben, die allen Ihren Auto-Scaling-Gruppen zugewiesen sind.

```
Get-ASNotificationConfiguration
```

- Einzelheiten zur API finden Sie unter [DescribeNotificationConfigurations AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribePolicies** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribePolicies` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie die Skalierungsrichtlinien einer angegebenen Gruppe

In diesem Beispiel wird die Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-policies \  
  --auto-scaling-group-name my-asg
```

Ausgabe:

```
{  
  "ScalingPolicies": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "PolicyName": "alb1000-target-tracking-scaling-policy",  
      "PolicyARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scalingPolicy:3065d9c8-9969-4bec-  
bb6a-3fbe5550fde6:autoScalingGroupName/my-asg:policyName/alb1000-target-tracking-  
scaling-policy",  
      "PolicyType": "TargetTrackingScaling",
```

```

    "StepAdjustments": [],
    "Alarms": [
      {
        "AlarmName": "TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196",
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196"
      },
      {
        "AlarmName": "TargetTracking-my-asg-AlarmLow-f96f899d-
b8e7-4d09-a010-c1aaa35da296",
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-a010-
c1aaa35da296"
      }
    ],
    "TargetTrackingConfiguration": {
      "PredefinedMetricSpecification": {
        "PredefinedMetricType": "ALBRequestCountPerTarget",
        "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-
alb-target-group/943f017f100becff"
      },
      "TargetValue": 1000.0,
      "DisableScaleIn": false
    },
    "Enabled": true
  },
  {
    "AutoScalingGroupName": "my-asg",
    "PolicyName": "cpu40-target-tracking-scaling-policy",
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:5fd26f71-39d4-4690-82a9-
b8515c45cdde:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-
scaling-policy",
    "PolicyType": "TargetTrackingScaling",
    "StepAdjustments": [],
    "Alarms": [
      {
        "AlarmName": "TargetTracking-my-asg-
AlarmHigh-139f9789-37b9-42ad-bea5-b5b147d7f473",
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-139f9789-37b9-42ad-
bea5-b5b147d7f473"
      }
    ]
  }

```

```

        },
        {
            "AlarmName": "TargetTracking-my-asg-AlarmLow-bd681c67-
fc18-4c56-8468-fb8e413009c9",
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-bd681c67-fc18-4c56-8468-
fb8e413009c9"
        }
    ],
    "TargetTrackingConfiguration": {
        "PredefinedMetricSpecification": {
            "PredefinedMetricType": "ASGAverageCPUUtilization"
        },
        "TargetValue": 40.0,
        "DisableScaleIn": false
    },
    "Enabled": true
}
]
}

```

Weitere Informationen finden Sie unter [Dynamische Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So beschreiben Sie die Skalierungsrichtlinien eines bestimmten Namens

Verwenden Sie die `--policy-names`-Option, um bestimmte Skalierungsrichtlinien zurückzugeben.

```

aws autoscaling describe-policies \
  --auto-scaling-group-name my-asg \
  --policy-names cpu40-target-tracking-scaling-policy

```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter [Dynamische Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So beschreiben Sie eine Reihe von Skalierungsrichtlinien

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von Richtlinien zurückzugeben.

```
aws autoscaling describe-policies \  
  --auto-scaling-group-name my-asg \  
  --max-items 1
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Wenn die Ausgabe ein NextToken-Feld enthält, verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf, um die zusätzlichen Richtlinien abzurufen.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg --starting-  
token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter [Dynamische Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribePolicies](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Ausgabe:

```
AdjustmentType      : ChangeInCapacity  
Alarms              : {}  
AutoScalingGroupName : my-asg  
Cooldown            : 0  
EstimatedInstanceWarmup : 0  
MetricAggregationType :  
MinAdjustmentMagnitude : 0  
MinAdjustmentStep    : 0  
PolicyARN           : arn:aws:auto-scaling:us-  
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef  
                    : autoScalingGroupName/my-asg:policyName/myScaleInPolicy
```

```

PolicyName      : myScaleInPolicy
PolicyType      : SimpleScaling
ScalingAdjustment : -1
StepAdjustments : {}

```

Beispiel 2: In diesem Beispiel werden die angegebenen Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```

Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
"myScaleInPolicy")

```

Beispiel 3: In diesem Beispiel werden alle Richtlinien für all Ihre Auto-Scaling-Gruppen beschrieben.

```

Get-ASPolicy

```

- Einzelheiten zur API finden Sie unter [DescribePolicies AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```

Get-ASPolicy -AutoScalingGroupName my-asg

```

Ausgabe:

```

AdjustmentType      : ChangeInCapacity
Alarms              : {}
AutoScalingGroupName : my-asg
Cooldown            : 0
EstimatedInstanceWarmup : 0
MetricAggregationType :
MinAdjustmentMagnitude : 0
MinAdjustmentStep    : 0
PolicyARN           : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                    :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName          : myScaleInPolicy
PolicyType          : SimpleScaling

```

```
ScalingAdjustment      : -1
StepAdjustments        : {}
```

Beispiel 2: In diesem Beispiel werden die angegebenen Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
"myScaleInPolicy")
```

Beispiel 3: In diesem Beispiel werden alle Richtlinien für all Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASPolicy
```

- Einzelheiten zur API finden Sie unter [DescribePolicies AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **DescribeScalingActivities** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScalingActivities` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
/// Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
public async Task<List<Activity>> DescribeScalingActivitiesAsync(
    string groupName)
{
    var activities = new List<Activity>();
    var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
    {
        AutoScalingGroupName = groupName,
        MaxRecords = 10,
    };

    var response = await
_amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
    if (response.Activities != null)
    {
        activities = response.Activities;
    }
    return activities;
}
```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
    Aws::String nextToken; // Used for pagination;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
            autoScalingClient.DescribeScalingActivities(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                outcome.GetResult().GetActivities();
            allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    } while (!nextToken.empty());

    std::cout << "Found " << allActivities.size() << " activities."
        << std::endl;
    std::cout << "Activities are ordered with the most recent first."
        << std::endl;
    for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
        std::cout << activity.GetDescription() << std::endl;
        std::cout << activity.GetDetails() << std::endl;
    }

```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie Skalierungsaktivitäten für die angegebene Gruppe

In diesem Beispiel werden die Skalierungsaktivitäten für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-scaling-activities \  
  --auto-scaling-group-name my-asg
```

Ausgabe:

```
{  
  "Activities": [  
    {  
      "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",  
      "Description": "Launching a new EC2 instance: i-0d44425630326060f",  
      "AutoScalingGroupName": "my-asg",  
      "Cause": "At 2020-10-30T19:35:51Z a user request update of  
AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the  
desired capacity from 0 to 16. At 2020-10-30T19:36:07Z an instance was started  
in response to a difference between desired and actual capacity, increasing the  
capacity from 0 to 16.",  
      "StartTime": "2020-10-30T19:36:09.766Z",  
      "EndTime": "2020-10-30T19:36:41Z",  
      "StatusCode": "Successful",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":  
\"us-west-2b\"}"  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Verifizieren einer Skalierungsaktivität für eine Auto-Scaling-Gruppe](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So beschreiben Sie die Skalierungsaktivitäten für eine gelöschte Gruppe

Fügen Sie die `--include-deleted-groups`-Option hinzu, um Skalierungsaktivitäten nach dem Löschen der Auto-Scaling-Gruppe zu beschreiben.

```
aws autoscaling describe-scaling-activities \  
  --auto-scaling-group-name my-asg \  
  --include-deleted-groups
```

Ausgabe:

```
{  
  "Activities": [  
    {  
      "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",  
      "Description": "Launching a new EC2 instance. Status Reason: Your Spot request price of 0.001 is lower than the minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance failed.",  
      "AutoScalingGroupName": "my-asg",  
      "Cause": "At 2021-01-13T20:47:24Z a user request update of AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 0 to 3. At 2021-01-13T20:47:27Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 3.",  
      "StartTime": "2021-01-13T20:47:30.094Z",  
      "EndTime": "2021-01-13T20:47:30Z",  
      "StatusCode": "Failed",  
      "StatusMessage": "Your Spot request price of 0.001 is lower than the minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance failed.",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2b\"}",  
      "AutoScalingGroupState": "Deleted",  
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:283179a2-f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Fehlerbehebung bei Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So beschreiben Sie eine bestimmte Anzahl von Skalierungsaktivitäten

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von Aktivitäten zurückzugeben.

```
aws autoscaling describe-scaling-activities \  
  --max-items 1
```

Ausgabe:

```
{  
  "Activities": [  
    {  
      "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",  
      "Description": "Launching a new EC2 instance: i-0d44425630326060f",  
      "AutoScalingGroupName": "my-asg",  
      "Cause": "At 2020-10-30T19:35:51Z a user request update of  
AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the  
desired capacity from 0 to 16. At 2020-10-30T19:36:07Z an instance was started  
in response to a difference between desired and actual capacity, increasing the  
capacity from 0 to 16.",  
      "StartTime": "2020-10-30T19:36:09.766Z",  
      "EndTime": "2020-10-30T19:36:41Z",  
      "StatusCode": "Successful",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":  
\"us-west-2b\"}"  
    }  
  ]  
}
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere Aktivitäten. Verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt, um die zusätzlichen Aktivitäten abzurufen.

```
aws autoscaling describe-scaling-activities \  
  --starting-token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter [Verifizieren einer Skalierungsaktivität für eine Auto-Scaling-Gruppe](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response = autoScalingClient
            .describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity : activities) {
            System.out.println("The activity Id is " +
activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
                ${group.healthCheckType}")
        }
    }
}
```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Ausgabe:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
group desired capacity changing the desired
                        capacity from 1 to 2. At 2015-11-22T15:45:34Z an instance
was started in response to a difference
                        between desired and actual capacity, increasing the
capacity from 1 to 2.
```

```

Description      : Launching a new EC2 instance: i-26e715fc
Details          : {"Availability Zone":"us-west-2b","Subnet
  ID":"subnet-5264e837"}
EndTime         : 11/22/2015 7:46:09 AM
Progress        : 100
StartTime       : 11/22/2015 7:45:35 AM
StatusCode      : Successful
StatusMessage   :

ActivityId      : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause           : At 2015-11-20T22:57:53Z a user request created an
  AutoScalingGroup changing the desired capacity
                  from 0 to 1. At 2015-11-20T22:57:58Z an instance was
  started in response to a difference betwe
                  en desired and actual capacity, increasing the capacity
  from 0 to 1.
Description     : Launching a new EC2 instance: i-93633f9b
Details         : {"Availability Zone":"us-west-2b","Subnet
  ID":"subnet-5264e837"}
EndTime        : 11/20/2015 2:58:32 PM
Progress       : 100
StartTime      : 11/20/2015 2:57:59 PM
StatusCode     : Successful
StatusMessage  :

```

Beispiel 2: In diesem Beispiel wird die angegebene Skalierungsaktivität beschrieben.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Beispiel 3: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für all Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScalingActivity
```

- Einzelheiten zur API finden Sie unter [DescribeScalingActivities AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Ausgabe:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
                      group desired capacity changing the desired
                      capacity from 1 to 2. At 2015-11-22T15:45:34Z an instance
                      was started in response to a difference
                      between desired and actual capacity, increasing the
                      capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime             : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime           : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage       :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
                      AutoScalingGroup changing the desired capacity
                      from 0 to 1. At 2015-11-20T22:57:58Z an instance was
                      started in response to a difference betwe
                      en desired and actual capacity, increasing the capacity
                      from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime             : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime           : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage       :
```

Beispiel 2: In diesem Beispiel wird die angegebene Skalierungsaktivität beschrieben.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Beispiel 3: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für all Ihre Auto-Scaling-Gruppen beschrieben.

Get-ASScalingActivity

- Einzelheiten zur API finden Sie unter [DescribeScalingActivities AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def describe_scaling_activities(self, group_name: str) -> List[Dict[str,
Any]]:
        """
        Gets information about scaling activities for the group. Scaling
        activities
        are things like instances stopping or starting in response to user
        requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: A list of dictionaries representing the scaling activities for
        the
                group, ordered with the most recent activity first.
```

```
        :raises ClientError: If there is an error describing the scaling
activities.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_scaling_activities"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupName=group_name)
            activities = []
            for response in response_iterator:
                activities.extend(response.get("Activities", []))

            logger.info(
                f"Successfully described scaling activities for group
'{{group_name}}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe scaling activities for group '{{group_name}}'.
Error code: {{error_code}}, Message: {{err.response['Error']['Message']}}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{{group_name}}'. "
                    "Please try again later."
                )
                raise
            else:
                return activities
```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
    let group = self
        .autoscaling
        .describe_auto_scaling_groups()
        .auto_scaling_group_names(self.auto_scaling_group_name.clone())
        .send()
        .await
        .map(|s| {
            s.auto_scaling_groups()
                .iter()
                .map(|s| {
                    format!(
                        "{}: {}",
                        s.auto_scaling_group_name().unwrap_or("Unknown"),
                        s.status().unwrap_or("Unknown")
                    )
                })
                .collect::
```

```
// Bonus: use CloudWatch API to get and show some metrics collected for
the group.
// CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
// CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
let activities = self
    .autoscaling
    .describe_scaling_activities()
    .auto_scaling_group_name(self.auto_scaling_group_name.clone())
    .into_paginator()
    .items()
    .send()
    .collect::<Result<Vec<_>, _>>()
    .await
    .map_err(|e| {
        anyhow!(
            "There was an error retrieving scaling activities: {}",
            DisplayErrorContext(&e)
        )
    });

AutoScalingScenarioDescription {
    group,
    instances,
    activities,
}
}
```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
  DATA(lo_output) = ao_asc->describescalingactivities(
    iv_autoscalinggroupname = iv_group_name ).

  ot_output = lo_output->get_activities( ).

  MESSAGE 'Scaling activities retrieved successfully' TYPE 'I'.

CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
  RAISE EXCEPTION lo_contention.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
  RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeScalingProcessTypes** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScalingProcessTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Prozesstypen

In diesem Beispiel werden die verfügbaren Prozesstypen beschrieben.

```
aws autoscaling describe-scaling-process-types
```

Ausgabe:

```
{
```

```
"Processes": [  
  {  
    "ProcessName": "AZRebalance"  
  },  
  {  
    "ProcessName": "AddToLoadBalancer"  
  },  
  {  
    "ProcessName": "AlarmNotification"  
  },  
  {  
    "ProcessName": "HealthCheck"  
  },  
  {  
    "ProcessName": "InstanceRefresh"  
  },  
  {  
    "ProcessName": "Launch"  
  },  
  {  
    "ProcessName": "ReplaceUnhealthy"  
  },  
  {  
    "ProcessName": "ScheduledActions"  
  },  
  {  
    "ProcessName": "Terminate"  
  }  
]  
}
```

Weitere Informationen erhalten Sie unter [Aus- und Fortsetzen von Skalierungsprozessen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeScalingProcessTypes](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Prozesstypen aufgelistet, die von Auto Scaling unterstützt werden.

```
Get-ASScalingProcessType
```

Ausgabe:

```
ProcessName
-----
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Einzelheiten zur API finden Sie unter [DescribeScalingProcessTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Prozesstypen aufgelistet, die von Auto Scaling unterstützt werden.

```
Get-ASScalingProcessType
```

Ausgabe:

```
ProcessName
-----
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Einzelheiten zur API finden Sie unter [DescribeScalingProcessTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeScheduledActions** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScheduledActions` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie alle geplanten Aktionen

In diesem Beispiel werden alle Ihre geplanten Aktionen beschrieben.

```
aws autoscaling describe-scheduled-actions
```

Ausgabe:

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}
```

Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So beschreiben Sie geplante Aktionen für die angegebene Gruppe

Verwenden Sie die `--auto-scaling-group-name`-Option, um die geplanten Aktionen für eine bestimmte Auto-Scaling-Gruppe zu beschreiben.

```
aws autoscaling describe-scheduled-actions \  
  --auto-scaling-group-name my-asg
```

Ausgabe:

```
{  
  "ScheduledUpdateGroupActions": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "ScheduledActionName": "my-recurring-action",  
      "Recurrence": "30 0 1 1,6,12 *",  
      "ScheduledActionARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-  
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-  
action",  
      "StartTime": "2023-12-01T04:00:00Z",  
      "Time": "2023-12-01T04:00:00Z",  
      "MinSize": 1,  
      "MaxSize": 6,  
      "DesiredCapacity": 4,  
      "TimeZone": "America/New_York"  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So beschreiben Sie die angegebene geplante Aktion

Verwenden Sie die `--scheduled-action-names`-Option, um eine bestimmte geplante Aktion zu beschreiben.

```
aws autoscaling describe-scheduled-actions \  
  --scheduled-action-names my-recurring-action
```

Ausgabe:

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}
```

Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 4: So beschreiben Sie geplante Aktionen mit einer bestimmten Startzeit

Verwenden Sie die `--start-time`-Option, um die geplanten Aktionen zu beschreiben, die zu einer bestimmten Zeit beginnen.

```
aws autoscaling describe-scheduled-actions \
  --start-time "2023-12-01T04:00:00Z"
```

Ausgabe:

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
```

```

b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
    "StartTime": "2023-12-01T04:00:00Z",
    "Time": "2023-12-01T04:00:00Z",
    "MinSize": 1,
    "MaxSize": 6,
    "DesiredCapacity": 4,
    "TimeZone": "America/New_York"
  }
]
}

```

Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 5: So beschreiben Sie geplante Aktionen, die zu einem bestimmten Zeitpunkt enden

Verwenden Sie die `--end-time`-Option, um die geplanten Aktionen zu beschreiben, die zu einem bestimmten Zeitpunkt enden.

```

aws autoscaling describe-scheduled-actions \
  --end-time "2023-12-01T04:00:00Z"

```

Ausgabe:

```

{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}

```

```
}
```

Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 6: So beschreiben Sie eine bestimmte Anzahl von geplanten Aktionen

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von geplanten Aktionen zurückzugeben.

```
aws autoscaling describe-scheduled-actions \  
  --auto-scaling-group-name my-asg \  
  --max-items 1
```

Ausgabe:

```
{  
  "ScheduledUpdateGroupActions": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "ScheduledActionName": "my-recurring-action",  
      "Recurrence": "30 0 1 1,6,12 *",  
      "ScheduledActionARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-  
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-  
action",  
      "StartTime": "2023-12-01T04:00:00Z",  
      "Time": "2023-12-01T04:00:00Z",  
      "MinSize": 1,  
      "MaxSize": 6,  
      "DesiredCapacity": 4,  
      "TimeZone": "America/New_York"  
    }  
  ]  
}
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere geplante Aktionen.

Verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt, um die zusätzlichen Aktivitäten abzurufen.

```
aws autoscaling describe-scheduled-actions \  
  --starting-token
```

```
--auto-scaling-group-name my-asg \  
--starting-token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeScheduledActions](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die geplanten Skalierungsaktionen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Ausgabe:

```
AutoScalingGroupName : my-asg  
DesiredCapacity      : 10  
EndTime              :  
MaxSize              :  
MinSize              :  
Recurrence           :  
ScheduledActionARN   : arn:aws:autoscaling:us-  
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7  
2c3af3a4d6:autoScalingGroupName/my-  
asg:scheduledActionName/myScheduledAction  
ScheduledActionName  : myScheduledAction  
StartTime             : 11/30/2015 8:00:00 AM  
Time                  : 11/30/2015 8:00:00 AM
```

Beispiel 2: In diesem Beispiel werden die angegebenen geplanten Skalierungsaktionen beschrieben.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",  
"myScheduledScaleIn")
```

Beispiel 3: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit beginnen.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Beispiel 4: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit enden.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Beispiel 5: In diesem Beispiel werden die geplanten Skalierungsaktionen für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScheduledAction
```

- Einzelheiten zur API finden Sie unter [DescribeScheduledActions AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die geplanten Skalierungsaktionen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Ausgabe:

```
AutoScalingGroupName : my-asg
DesiredCapacity       : 10
EndTime               :
MaxSize               :
MinSize               :
Recurrence             :
ScheduledActionARN    : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                      2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
ScheduledActionName   : myScheduledAction
StartTime              : 11/30/2015 8:00:00 AM
Time                  : 11/30/2015 8:00:00 AM
```

Beispiel 2: In diesem Beispiel werden die angegebenen geplanten Skalierungsaktionen beschrieben.

```
Get-ASScheduledAction -ScheduledActionName @"myScheduledScaleOut",  
"myScheduledScaleIn")
```

Beispiel 3: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit beginnen.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Beispiel 4: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit enden.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Beispiel 5: In diesem Beispiel werden die geplanten Skalierungsaktionen für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScheduledAction
```

- Einzelheiten zur API finden Sie unter [DescribeScheduledActions AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeTags** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTags` verwendet wird.

CLI

AWS CLI

So beschreiben Sie alle Tags

In diesem Beispiel werden alle Ihre Tags beschrieben.

```
aws autoscaling describe-tags
```

Ausgabe:

```
{
  "Tags": [
    {
      "ResourceType": "auto-scaling-group",
      "ResourceId": "my-asg",
      "PropagateAtLaunch": true,
      "Value": "Research",
      "Key": "Dept"
    },
    {
      "ResourceType": "auto-scaling-group",
      "ResourceId": "my-asg",
      "PropagateAtLaunch": true,
      "Value": "WebServer",
      "Key": "Role"
    }
  ]
}
```

Weitere Informationen finden Sie unter [Markieren von Auto Scaling-Gruppen und Instances](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 2: So beschreiben Sie Tags für eine bestimmte Gruppe

Verwenden Sie die `--filters`-Option, um Tags für eine bestimmte Auto-Scaling-Gruppe zu beschreiben.

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

Weitere Informationen finden Sie unter [Markieren von Auto Scaling-Gruppen und Instances](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 3: So beschreiben Sie die angegebene Anzahl von Tags

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von Tags zurückzugeben.

```
aws autoscaling describe-tags \  
  --max-items 1
```

Wenn die Ausgabe ein NextToken-Feld enthält, gibt es weitere Tags. Verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt, um die zusätzlichen Tags zu erhalten.

```
aws autoscaling describe-tags \  
  --filters Name=auto-scaling-group,Values=my-asg \  
  --starting-token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter [Markieren von Auto Scaling-Gruppen und Instances](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeTags](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Tags mit dem Schlüsselwert „myTag“ oder „myTag2“ beschrieben. Die möglichen Werte für den Filternamen sind `auto-scaling-group`, `'Schlüssel'`, `'Wert'` und `propagate-at-launch`. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Ausgabe:

```
Key           : myTag2  
PropagateAtLaunch : True  
ResourceId    : my-asg  
ResourceType  : auto-scaling-group  
Value        : myTagValue2  
  
Key           : myTag  
PropagateAtLaunch : True  
ResourceId    : my-asg  
ResourceType  : auto-scaling-group  
Value        : myTagValue
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie `New-Object` verwenden, um den Filter für den Filter-Parameter zu erstellen.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Beispiel 3: In diesem Beispiel werden alle Tags für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASTag
```

- Einzelheiten zur API finden Sie unter [DescribeTags AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Tags mit dem Schlüsselwert „myTag“ oder „myTag2“ beschrieben. Die möglichen Werte für den Filternamen sind 'auto-scaling-group', 'Schlüssel', 'Wert' und propagate-at-launch '. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Ausgabe:

```
Key           : myTag2
PropagateAtLaunch : True
ResourceId    : my-asg
ResourceType  : auto-scaling-group
Value        : myTagValue2

Key           : myTag
PropagateAtLaunch : True
ResourceId    : my-asg
ResourceType  : auto-scaling-group
Value        : myTagValue
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um den Filter für den Filter-Parameter zu erstellen.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Beispiel 3: In diesem Beispiel werden alle Tags für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASTag
```

- Einzelheiten zur API finden Sie unter [DescribeTags AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DescribeTerminationPolicyTypes** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTerminationPolicyTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Typen von Beendigungsrichtlinien

In diesem Beispiel werden die verfügbaren Arten von Beendigungsrichtlinien beschrieben.

```
aws autoscaling describe-termination-policy-types
```

Ausgabe:

```
{
  "TerminationPolicyTypes": [
    "AllocationStrategy",
    "ClosestToNextInstanceHour",
    "Default",
```

```
    "NewestInstance",  
    "OldestInstance",  
    "OldestLaunchConfiguration",  
    "OldestLaunchTemplate"  
  ]  
}
```

Weitere Informationen finden Sie unter [Steuern, welche Auto Scaling-Instances bei der horizontalen Skalierung nach unten beendet werden](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DescribeTerminationPolicyTypes](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Beendigungsrichtlinien aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASTerminationPolicyType
```

Ausgabe:

```
ClosestToNextInstanceHour  
Default  
NewestInstance  
OldestInstance  
OldestLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter [DescribeTerminationPolicyTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Beendigungsrichtlinien aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASTerminationPolicyType
```

Ausgabe:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter [DescribeTerminationPolicyTypes AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DetachInstances** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DetachInstances verwendet wird.

CLI

AWS CLI

So trennen Sie eine Instance von einer Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt.

```
aws autoscaling detach-instances \
  --instance-ids i-030017cfa84b20135 \
  --auto-scaling-group-name my-asg \
  --should-decrement-desired-capacity
```

Ausgabe:

```
{
  "Activities": [
    {
      "ActivityId": "5091cb52-547a-47ce-a236-c9ccbc2cb2c9",
      "AutoScalingGroupName": "my-asg",
      "Description": "Detaching EC2 instance: i-030017cfa84b20135",
      "Cause": "At 2020-10-31T17:35:04Z instance i-030017cfa84b20135 was detached in response to a user request, shrinking the capacity from 2 to 1.",
    }
  ]
}
```

```

        "StartTime": "2020-04-12T15:02:16.179Z",
        "StatusCode": "InProgress",
        "Progress": 50,
        "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\\\"us-west-2c\\\"}"
    }
]
}

```

- Einzelheiten zur API finden Sie [DetachInstances](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```

ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
                      in response to a user request, shrinking
                      the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress              : 50
StartTime             : 11/20/2015 2:34:59 PM
StatusCode            : InProgress
StatusMessage        :

```

Beispiel 2: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause               : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
                    in response to a user request.
Description         : Detaching EC2 instance: i-7bf746a2
Details             : {"Availability Zone":"us-west-2b","Subnet
                    ID":"subnet-5264e837"}
EndTime            :
Progress            : 50
StartTime           : 11/20/2015 2:34:59 PM
StatusCode          : InProgress
StatusMessage       :
```

- Einzelheiten zur API finden Sie unter [DetachInstances AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause               : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
                    in response to a user request, shrinking
                    the capacity from 2 to 1.
Description         : Detaching EC2 instance: i-93633f9b
Details             : {"Availability Zone":"us-west-2b","Subnet
                    ID":"subnet-5264e837"}
EndTime            :
```

```
Progress           : 50
StartTime          : 11/20/2015 2:34:59 PM
StatusCode         : InProgress
StatusMessage      :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId         : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause              : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
                   in response to a user request.
Description        : Detaching EC2 instance: i-7bf746a2
Details            : {"Availability Zone":"us-west-2b","Subnet
                   ID":"subnet-5264e837"}
EndTime           :
Progress          : 50
StartTime         : 11/20/2015 2:34:59 PM
StatusCode        : InProgress
StatusMessage     :
```

- Einzelheiten zur API finden Sie unter [DetachInstances AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DetachLoadBalancers** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachLoadBalancers` verwendet wird.

CLI

AWS CLI

So trennen Sie einen Classic Load Balancer von einer Auto-Scaling-Gruppe

In diesem Beispiel wird der angegebene Classic Load Balancer von der spezifizierten Auto-Scaling-Gruppe getrennt.

```
aws autoscaling detach-load-balancers \  
  --load-balancer-names my-load-balancer \  
  --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Anhängen eines Load Balancers an Ihre Auto-Scaling-Gruppe](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [DetachLoadBalancers](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer von der spezifizierten Auto-Scaling-Gruppe getrennt.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [DetachLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz \(V4\)](#).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer von der spezifizierten Auto-Scaling-Gruppe getrennt.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [DetachLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **DisableMetricsCollection** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisableMetricsCollection` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Disable the collection of metric data for an Amazon EC2 Auto Scaling
/// group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public async Task<bool> DisableMetricsCollectionAsync(string groupName)
{
    var request = new DisableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
    };

    var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
request.SetAutoScalingGroupName(groupName);

Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
    autoScalingClient.DisableMetricsCollection(request);

if (outcome.IsSuccess()) {
    std::cout << "Metrics collection has been disabled." << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
        << outcome.GetError().GetMessage()
        << std::endl;
}
}
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So deaktivieren Sie die Erfassung von Metriken für Auto-Scaling-Gruppen

In diesem Beispiel wird die Erfassung der `GroupDesiredCapacity`-Metrik für die angegebene Auto-Scaling-Gruppe deaktiviert.

```
aws autoscaling disable-metrics-collection \  
  --auto-scaling-group-name my-asg \  
  --metrics GroupDesiredCapacity
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [CloudWatch Monitoring-Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public static void disableMetricsCollection(AutoScalingClient  
autoScalingClient, String groupName) {  
    try {  
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =  
DisableMetricsCollectionRequest.builder()  
            .autoScalingGroupName(groupName)  
            .metrics("GroupMaxSize")  
            .build();  
  
autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
```

```
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function disableMetricsCollection($autoScalingGroupName)
{
    return $this->autoScalingClient->disableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die spezifizierten Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
@("GroupMinSize", "GroupMaxSize")
```

Beispiel 2: In diesem Beispiel wird die Überwachung aller Metriken für die angegebene Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [DisableMetricsCollection AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die spezifizierten Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric  
@("GroupMinSize", "GroupMaxSize")
```

Beispiel 2: In diesem Beispiel wird die Überwachung aller Metriken für die angegebene Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [DisableMetricsCollection AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def disable_metrics(self, group_name: str) -> Dict[str, Any]:  
        """  
        Stops CloudWatch metric collection for the Auto Scaling group.
```

```
    :param group_name: The name of the group.
    :return: A dictionary with the response from disabling the metrics
collection.
    :raises ClientError: If there is an error disabling metrics collection.
    """
    try:
        response = self.autoscaling_client.disable_metrics_collection(
            AutoScalingGroupName=group_name
        )
        logger.info(
            f"Successfully disabled metrics collection for group
'{group_name}'."
        )
        return response
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(
            f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
        )

        if error_code == "ResourceContentionFault":
            logger.error(
                f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                "Please try again later."
            )
            raise
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
let _ = self
    .autoscaling
    .disable_metrics_collection()
    .auto_scaling_group_name(self.auto_scaling_group_name.clone())
    .send()
    .await;
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
    ao_asc->disablemetricscollection(
        iv_autoscalinggroupname = iv_group_name ).
```

```
MESSAGE 'Metrics collection disabled successfully' TYPE 'I'.

CATCH /aws1/cx_ascsresrcontionfault INTO DATA(lo_contention).
  RAISE EXCEPTION lo_contention.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
  RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **EnableMetricsCollection** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableMetricsCollection` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Enable the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
```

```
public async Task<bool> EnableMetricsCollectionAsync(string groupName)
{
    var listMetrics = new List<string>
    {
        "GroupMaxSize",
    };

    var collectionRequest = new EnableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
        Metrics = listMetrics,
        Granularity = "1Minute",
    };

    var response = await
        _amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
```

```
request.SetAutoScalingGroupName(groupName);

request.AddMetrics("GroupMinSize");
request.AddMetrics("GroupMaxSize");
request.AddMetrics("GroupDesiredCapacity");
request.AddMetrics("GroupInServiceInstances");
request.AddMetrics("GroupTotalInstances");
request.SetGranularity("1Minute");

Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
    autoScalingClient.EnableMetricsCollection(request);
if (outcome.IsSuccess()) {
    std::cout << "Auto Scaling metrics have been enabled."
              << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
}
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So aktivieren Sie die Erfassung von Auto-Scaling-Gruppen

In diesem Beispiel wird die Datenerfassung für die angegebene Auto-Scaling-Gruppe aktiviert.

```
aws autoscaling enable-metrics-collection \
  --auto-scaling-group-name my-asg \
  --granularity "1Minute"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [CloudWatch Monitoring-Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

Beispiel 2: So sammeln Sie Daten für die angegebene Metrik einer Auto-Scaling-Gruppe

Verwenden Sie die `--metrics`-Option, um Daten für eine bestimmte Metrik zu sammeln.

```
aws autoscaling enable-metrics-collection \  
  --auto-scaling-group-name my-asg \  
  --metrics GroupDesiredCapacity --granularity "1Minute"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [CloudWatch Monitoring-Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public static void enableMetricsCollection(AutoScalingClient  
autoScalingClient, String groupName) {  
    try {  
        EnableMetricsCollectionRequest collectionRequest =  
EnableMetricsCollectionRequest.builder()  
            .autoScalingGroupName(groupName)  
            .metrics("GroupMaxSize")  
            .granularity("1Minute")  
            .build();  
  
        autoScalingClient.enableMetricsCollection(collectionRequest);  
        System.out.println("The enable metrics collection operation was  
successful");  
  
    } catch (AutoScalingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
    }  
}
```

```
        System.exit(1);
    }
}
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function enableMetricsCollection($autoScalingGroupName, $granularity)
{
    return $this->autoScalingClient->enableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'Granularity' => $granularity,
    ]);
}
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die angegebene Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection -Metric @"GroupMinSize", "GroupMaxSize" -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Beispiel 2: In diesem Beispiel wird eine Überwachung aller Metriken der angegebenen Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Einzelheiten zur API finden Sie unter [EnableMetricsCollection AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die angegebene Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection -Metric @"GroupMinSize", "GroupMaxSize" -  
AutoScalingGroupName my-asg -Granularity 1Minute
```

Beispiel 2: In diesem Beispiel wird eine Überwachung aller Metriken der angegebenen Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Einzelheiten zur API finden Sie unter [EnableMetricsCollection AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str,  
Any]:  
        """
```

Enables CloudWatch metric collection for Amazon EC2 Auto Scaling activities.

```
:param group_name: The name of the group to enable.
:param metrics: A list of metrics to collect.
:return: A dictionary with the response from enabling the metrics
collection.
:raises ClientError: If there is an error enabling metrics collection.
"""
try:
    response = self.autoscaling_client.enable_metrics_collection(
        AutoScalingGroupName=group_name, Metrics=metrics,
Granularity="1Minute"
    )
    logger.info(
        f"Successfully enabled metrics for Auto Scaling group
'{group_name}'."
    )

except ClientError as err:
    error_code = err.response["Error"]["Code"]
    logger.error(
        f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
    )

    if error_code == "ResourceContentionFault":
        logger.error(
            f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
            "Please try again later."
        )
    elif error_code == "InvalidParameterCombination":
        logger.error(
            f"The combination of parameters provided for enabling metrics
on '{group_name}' is not valid. "
            "Please check the parameters and try again."
        )
    raise
else:
    return response
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
let enable_metrics_collection = autoscaling
    .enable_metrics_collection()
    .auto_scaling_group_name(auto_scaling_group_name.as_str())
    .granularity("1Minute")
    .set_metrics(Some(vec![
        String::from("GroupMinSize"),
        String::from("GroupMaxSize"),
        String::from("GroupDesiredCapacity"),
        String::from("GroupInServiceInstances"),
        String::from("GroupTotalInstances"),
    ]))
    .send()
    .await;
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
" Example: iv_group_name = 'my-auto-scaling-group'
" Example: it_metrics contains list of metrics like 'GroupMinSize',
'GroupMaxSize', etc.
```

```
TRY.
```

```
ao_asc->enablemetricscollection(
  iv_autoscalinggroupname = iv_group_name
  it_metrics = it_metrics
  iv_granularity = '1Minute' ).
```

```
MESSAGE 'Metrics collection enabled successfully' TYPE 'I'.
```

```
CATCH /aws1/cx_asresrccontionfault INTO DATA(lo_contention).
```

```
RAISE EXCEPTION lo_contention.
```

```
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
```

```
RAISE EXCEPTION lo_generic_exception.
```

```
ENDTRY.
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **EnterStandby** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie EnterStandby verwendet wird.

CLI

AWS CLI

So verschieben Sie Instances in den Standby-Modus

In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt. Das ist nützlich, um eine Instance zu aktualisieren oder Fehler zu beheben, die sich derzeit im Service befindet.

```
aws autoscaling enter-standby \
```

```
--instance-ids i-061c63c5eb45f0416 \  
--auto-scaling-group-name my-asg \  
--should-decrement-desired-capacity
```

Ausgabe:

```
{  
  "Activities": [  
    {  
      "ActivityId": "ffa056b4-6ed3-41ba-ae7c-249dfae6eba1",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Moving EC2 instance to Standby: i-061c63c5eb45f0416",  
      "Cause": "At 2020-10-31T20:31:00Z instance i-061c63c5eb45f0416 was  
moved to standby in response to a user request, shrinking the capacity from 1 to  
0.",  
      "StartTime": "2020-10-31T20:31:00.949Z",  
      "StatusCode": "InProgress",  
      "Progress": 50,  
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":  
\"us-west-2c\"}"  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Instance-Lebenszyklus in Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [EnterStandby](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId          : e36a5a54-ced6-4df8-bd19-708e2a59a649
```

```

AutoScalingGroupName : my-asg
Cause                 : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
                       standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description           : Moving EC2 instance to Standby: i-95b8484f
Details               : {"Availability Zone":"us-west-2b","Subnet
                       ID":"subnet-5264e837"}
EndTime              :
Progress              : 50
StartTime             : 11/22/2015 7:48:06 AM
StatusCode            : InProgress
StatusMessage        :

```

Beispiel 2: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```

Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false

```

Ausgabe:

```

ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
                       standby in response to a user request.
Description           : Moving EC2 instance to Standby: i-95b8484f
Details               : {"Availability Zone":"us-west-2b","Subnet
                       ID":"subnet-5264e837"}
EndTime              :
Progress              : 50
StartTime             : 11/22/2015 7:48:06 AM
StatusCode            : InProgress
StatusMessage        :

```

- Einzelheiten zur API finden Sie unter [EnterStandby AWS -Tools für PowerShellCmdlet-Referenz \(V4\)](#).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
                      standby in response to a user request,
                      shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
                      standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter [EnterStandby AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **ExecutePolicy** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ExecutePolicy` verwendet wird.

CLI

AWS CLI

So führen Sie eine Skalierungsrichtlinie aus

In diesem Beispiel wird die Skalierungsrichtlinie mit dem Namen `my-step-scale-out-policy` für die angegebene Auto-Scaling-Gruppe ausgeführt.

```
aws autoscaling execute-policy \  
  --auto-scaling-group-name my-asg \  
  --policy-name my-step-scale-out-policy \  
  --metric-value 95 \  
  --breach-threshold 80
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Schrittweise und einfache Skalierungsrichtlinien](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [ExecutePolicy](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel führt die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe aus.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Beispiel 2: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe ausgeführt, nachdem die Wartezeit abgelaufen ist.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter [ExecutePolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel führt die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe aus.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Beispiel 2: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe ausgeführt, nachdem die Wartezeit abgelaufen ist.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter [ExecutePolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **ExitStandby** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ExitStandby` verwendet wird.

CLI

AWS CLI

So verschieben Sie Instances aus dem Standby-Modus

In diesem Beispiel wird die angegebenen Instance aus dem Standby-Modus verschoben.

```
aws autoscaling exit-standby \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg
```

Ausgabe:

```
{  
  "Activities": [  
    {  
      "ActivityId": "142928e1-a2dc-453a-9b24-b85ad6735928",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Moving EC2 instance out of Standby:  
i-061c63c5eb45f0416",  
      "Cause": "At 2020-10-31T20:32:50Z instance i-061c63c5eb45f0416 was  
moved out of standby in response to a user request, increasing the capacity from  
0 to 1.",  
      "StartTime": "2020-10-31T20:32:50.222Z",  
      "StatusCode": "PreInService",  
      "Progress": 30,  
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":  
\"us-west-2c\"}"  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [ExitStandby](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance aus dem Standby-Modus verschoben.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Ausgabe:

```

ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
                      of standby in response to a user
                      request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :

```

- Einzelheiten zur API finden Sie unter [ExitStandby AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance aus dem Standby-Modus verschoben.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Ausgabe:

```

ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
                      of standby in response to a user
                      request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :

```

- Einzelheiten zur API finden Sie unter [ExitStandby AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **PutLifecycleHook** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutLifecycleHook` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen Lebenszyklus-Hook

In diesem Beispiel wird ein Lebenszyklus-Hook erstellt, der bei allen neu gestarteten Instances mit einem Timeout von 4 800 Sekunden aufgerufen wird. Dies ist nützlich, um die Instanzen im Wartezustand zu halten, bis die Benutzerdatenskripts abgeschlossen sind, oder um eine AWS Lambda-Funktion mit aufzurufen. `EventBridge`

```
aws autoscaling put-lifecycle-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-hook-name my-launch-hook \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \  
  --heartbeat-timeout 4800
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Wenn ein Lebenszyklus-Hook mit demselben Namen bereits vorhanden ist, wird er durch den neuen Lebenszyklus-Hook überschrieben.

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So senden Sie eine E-Mail-Nachricht in Amazon SNS, die Sie über den Statuswechsel einer Instance informiert

In diesem Beispiel wird ein Lebenszyklus-Hook mit dem Amazon-SNS-Thema und der IAM-Rolle erstellt, die zum Empfangen von Benachrichtigungen beim Instance-Start verwendet werden sollen.

```
aws autoscaling put-lifecycle-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-hook-name my-launch-hook \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \  
  --heartbeat-timeout 4800
```

```
--lifecycle-hook-name my-launch-hook \  
--lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \  
--notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \  
--role-arn arn:aws:iam::123456789012:role/my-auto-scaling-role
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So veröffentlichen Sie eine Nachricht in einer Amazon-SQS-Warteschlange

In diesem Beispiel wird ein Lebenszyklus-Hook erstellt, der eine Nachricht mit Metadaten in der angegebenen Amazon-SQS-Warteschlange veröffentlicht.

```
aws autoscaling put-lifecycle-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-hook-name my-launch-hook \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \  
  --notification-target-arn arn:aws:sqs:us-west-2:123456789012:my-sqs-queue \  
  --role-arn arn:aws:iam::123456789012:role/my-notification-role \  
  --notification-metadata "SQS message metadata"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [PutLifecycleHook](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook zur spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
  "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -  
  NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN  
  "arn:aws:iam::123456789012:role/my-iam-role"
```

- Einzelheiten zur API finden Sie unter [PutLifecycleHook AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook zur spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
  "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
  "arn:aws:iam::123456789012:role/my-iam-role"
```

- Einzelheiten zur API finden Sie unter [PutLifecycleHook AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **PutNotificationConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie PutNotificationConfiguration verwendet wird.

CLI

AWS CLI

So fügen Sie eine neue Benachrichtigung hinzu

In diesem Beispiel wird die angegebene Benachrichtigung der spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling put-notification-configuration \
  --auto-scaling-group-name my-asg \
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
  --notification-type autoscaling:TEST_NOTIFICATION
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [PutNotificationConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten von EC2-Instances eine Benachrichtigung an das spezifizierte SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten oder Beenden von EC2-Instances eine Benachrichtigung an das angegebene SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
@("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Einzelheiten zur API finden Sie unter [PutNotificationConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten von EC2-Instances eine Benachrichtigung an das spezifizierte SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten oder Beenden von EC2-Instances eine Benachrichtigung an das angegebene SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
@("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Einzelheiten zur API finden Sie unter [PutNotificationConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **PutScalingPolicy** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie PutScalingPolicy verwendet wird.

CLI

AWS CLI

So fügen Sie einer Auto-Scaling-Gruppe eine Richtlinie zur Zielverfolgung hinzu

Im folgenden Beispiel für put-scaling-policy wird eine Skalierungsrichtlinie zur Zielverfolgung auf die angegebene Auto-Scaling-Gruppe angewendet. Die Ausgabe enthält die Namen ARNs und der beiden CloudWatch Alarmer, die in Ihrem Namen erstellt wurden. Wenn bereits eine Skalierungsrichtlinie mit demselben Namen vorhanden ist, wird diese durch die neue Skalierungsrichtlinie überschrieben.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
--policy-name alb1000-target-tracking-scaling-policy \
--policy-type TargetTrackingScaling \
--target-tracking-configuration file://config.json
```

Inhalt von config.json:

```
{
  "TargetValue": 1000.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ALBRequestCountPerTarget",
    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
```

```
}
}
```

Ausgabe:

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-
c665-4bfd-aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/alb1000-
target-tracking-scaling-policy",
  "Alarms": [
    {
      "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
      "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
    },
    {
      "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
      "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }
  ]
}
```

Weitere Beispiele finden Sie unter [Beispiel für Skalierungsrichtlinien für die AWS Befehlszeilenschnittstelle \(AWS CLI\)](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

- Einzelheiten zur API finden Sie [PutScalingPolicy](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie der spezifizierten Auto-Scaling-Gruppe hinzugefügt. Der angegebene Anpassungstyp bestimmt, wie der ScalingAdjustment Parameter interpretiert wird. Bei 'ChangeInCapacity' erhöht ein positiver Wert die Kapazität um die angegebene Anzahl von Instanzen und ein negativer Wert verringert die Kapazität um die angegebene Anzahl von Instanzen.

```
Write-AScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType  
"ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Ausgabe:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-  
e1d769fc24ef:autoScalingGroupName/my-asg  
:policyName/myScaleInPolicy
```

- Einzelheiten zur API finden Sie unter [PutScalingPolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie der spezifizierten Auto-Scaling-Gruppe hinzugefügt. Der angegebene Anpassungstyp bestimmt, wie der ScalingAdjustment Parameter interpretiert wird. Bei 'ChangeInCapacity' erhöht ein positiver Wert die Kapazität um die angegebene Anzahl von Instanzen und ein negativer Wert verringert die Kapazität um die angegebene Anzahl von Instanzen.

```
Write-AScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType  
"ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Ausgabe:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-  
e1d769fc24ef:autoScalingGroupName/my-asg  
:policyName/myScaleInPolicy
```

- Einzelheiten zur API finden Sie unter [PutScalingPolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **PutScheduledUpdateGroupAction** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie PutScheduledUpdateGroupAction verwendet wird.

CLI

AWS CLI

Beispiel 1: So fügen Sie einer Auto-Scaling-Gruppe eine geplante Aktion hinzu

In diesem Beispiel wird die angegebene geplante Aktion der angegebenen Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling put-scheduled-update-group-action \  
  --auto-scaling-group-name my-asg \  
  --scheduled-action-name my-scheduled-action \  
  --start-time "2023-05-12T08:00:00Z" \  
  --min-size 2 \  
  --max-size 6 \  
  --desired-capacity 4
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Wenn eine geplante Aktion mit demselben Namen bereits vorhanden ist, wird sie durch die neue geplante Aktion überschrieben.

Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So geben Sie einen wiederkehrenden Zeitplan an

In diesem Beispiel wird eine geplante Aktion erstellt, die nach einem wiederkehrenden Zeitplan skaliert wird und jedes Jahr am 1. Januar, Juni und Dezember um 00:30 Uhr ausgeführt wird.

```
aws autoscaling put-scheduled-update-group-action \  
  --auto-scaling-group-name my-asg \  
  --scheduled-action-name my-recurring-action \  
  --recurrence "30 0 1 1,6,12 *" \  
  --min-size 2 \  
  --max-size 6 \  
  --desired-capacity 4
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Wenn eine geplante Aktion mit demselben Namen bereits vorhanden ist, wird sie durch die neue geplante Aktion überschrieben.

Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [PutScheduledUpdateGroupAction](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine einmalig geplante Aktion erstellt oder aktualisiert, um die gewünschte Kapazität zur angegebenen Startzeit zu ändern.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- Einzelheiten zur API finden Sie unter [PutScheduledUpdateGroupAction AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine einmalig geplante Aktion erstellt oder aktualisiert, um die gewünschte Kapazität zur angegebenen Startzeit zu ändern.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- Einzelheiten zur API finden Sie unter [PutScheduledUpdateGroupAction AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **RecordLifecycleActionHeartbeat** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie RecordLifecycleActionHeartbeat verwendet wird.

CLI

AWS CLI

So zeichnen Sie den Heartbeat einer Lebenszyklusaktion auf

Dieses Beispiel zeichnet den Heartbeat für eine Lebenszyklusaktion auf, um die Instance im Status „Ausstehend“ zu halten.

```
aws autoscaling record-lifecycle-action-heartbeat \  
  --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Lebenszyklus-Hooks für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [RecordLifecycleActionHeartbeat](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Heartbeat für die angegebene Lebenszyklusaktion aufgezeichnet. Dadurch bleibt die Instance im Status „Ausstehend“, bis die benutzerdefinierte Aktion abgeschlossen ist.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter [RecordLifecycleActionHeartbeat AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Heartbeat für die angegebene Lebenszyklusaktion aufgezeichnet. Dadurch bleibt die Instance im Status „Ausstehend“, bis die benutzerdefinierte Aktion abgeschlossen ist.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter [RecordLifecycleActionHeartbeat AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Verwenden Sie diesen Service mit einem SDK AWS](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **ResumeProcesses** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie ResumeProcesses verwendet wird.

CLI

AWS CLI

So setzen Sie einen ausgesetzten Prozess fort

In diesem Beispiel wird der angegebene ausgesetzte Skalierungsprozess für die spezifizierte Auto-Scaling-Gruppe fortgesetzt.

```
aws autoscaling resume-processes \  
  --auto-scaling-group-name my-asg \  
  --scaling-processes AlarmNotification
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen erhalten Sie unter [Aus- und Fortsetzen von Skalierungsprozessen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [ResumeProcesses](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle ausgesetzten Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [ResumeProcesses AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle ausgesetzten Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [ResumeProcesses AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **SetDesiredCapacity** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetDesiredCapacity` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Set the desired capacity of an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="desiredCapacity">The desired capacity for the Auto
/// Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> SetDesiredCapacityAsync(
    string groupName,
    int desiredCapacity)
{
    var capacityRequest = new SetDesiredCapacityRequest
    {
        AutoScalingGroupName = groupName,
        DesiredCapacity = desiredCapacity,
    };

    var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
    Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
request.SetAutoScalingGroupName(groupName);
request.SetDesiredCapacity(2);

Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
    autoScalingClient.SetDesiredCapacity(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
               << outcome.GetError().GetMessage()
               << std::endl;
}
}
```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So geben Sie die gewünschte Kapazität für die Auto-Scaling-Gruppe an

In diesem Beispiel wird die gewünschte Kapazität für die angegebene Auto-Scaling-Gruppe angegeben.

```
aws autoscaling set-desired-capacity \  
  --auto-scaling-group-name my-asg \  
  --desired-capacity 2 \  
  --honor-cooldown
```

Wenn dieser Befehl erfolgreich war, kehrt er zum Prompt zurück.

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,  
String groupName) {  
    try {  
        SetDesiredCapacityRequest capacityRequest =  
SetDesiredCapacityRequest.builder()  
            .autoScalingGroupName(groupName)  
            .desiredCapacity(2)  
            .build();  
  
        autoScalingClient.setDesiredCapacity(capacityRequest);  
        System.out.println("You have set the DesiredCapacity to 2");  
  
    } catch (AutoScalingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
{
    return $this->autoScalingClient->setDesiredCapacity([
```

```
        'AutoScalingGroupName' => $autoScalingGroupName,  
        'DesiredCapacity' => $desiredCapacity,  
    ]);  
}
```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Beispiel 2: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben und gewartet, bis die Ruhephase abgeschlossen ist, bevor auf die neue Größe skaliert wird.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -  
HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter [SetDesiredCapacity AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Beispiel 2: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben und gewartet, bis die Ruhephase abgeschlossen ist, bevor auf die neue Größe skaliert wird.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -  
HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter [SetDesiredCapacity AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
        keep the
        number of running instances equal to the desired capacity.

        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
```

```

        f"Successfully set desired capacity of {capacity} for Auto
Scaling group '{group_name}'."
    )

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(
            f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
        )
        if error_code == "ScalingActivityInProgress":
            logger.error(
                f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'. "
                "Please wait for the activity to complete before attempting
to set the desired capacity."
            )
            logger.error(f"Full error:\n\t{err}")
            raise

```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
    // 7. SetDesiredCapacity: set desired capacity to 2.
    // Wait for a second instance to launch.
    let update_group = self
        .autoscaling

```

```

        .set_desired_capacity()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .desired_capacity(capacity)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            format!("Failed to update group to desired capacity
({capacity}))").as_str(),
            &err,
        ));
    }
    Ok(())
}

```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_capacity = 2

TRY.
    ao_asc->setdesiredcapacity(
        iv_autoscalinggroupname = iv_group_name
        iv_desiredcapacity = iv_capacity
        iv_honorcooldown = abap_false ).

    MESSAGE 'Desired capacity set successfully' TYPE 'I'.

CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).

```

```
RAISE EXCEPTION lo_activity_in_progress.  
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).  
RAISE EXCEPTION lo_generic_exception.  
ENDTRY.
```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **SetInstanceHealth** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie SetInstanceHealth verwendet wird.

CLI

AWS CLI

So geben Sie den Zustand einer Instance an

In diesem Beispiel wird der Zustand der angegebenen Instance auf Unhealthy gesetzt.

```
aws autoscaling set-instance-health \  
  --instance-id i-061c63c5eb45f0416 \  
  --health-status Unhealthy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [SetInstanceHealth](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerhaft“ gesetzt, wodurch sie außer Betrieb genommen wird. Auto Scaling beendet und ersetzt die Instance.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerfrei“ gesetzt, sodass sie weiterhin in Betrieb bleibt. Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -  
ShouldRespectGracePeriod $false
```

- Einzelheiten zur API finden Sie unter [SetInstanceHealth AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerhaft“ gesetzt, wodurch sie außer Betrieb genommen wird. Auto Scaling beendet und ersetzt die Instance.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerfrei“ gesetzt, sodass sie weiterhin in Betrieb bleibt. Legen Sie die Wartezeit für die Zustandsprüfung einer Auto-Scaling-Gruppe fest.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -  
ShouldRespectGracePeriod $false
```

- Einzelheiten zur API finden Sie unter [SetInstanceHealth AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **SetInstanceProtection** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetInstanceProtection` verwendet wird.

CLI

AWS CLI

Beispiel 1: So ändern Sie die Instance-Schutzeinstellung für eine Instance

In diesem Beispiel wird der Instance-Schutz für die angegebene Instance aktiviert.

```
aws autoscaling set-instance-protection \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg --protected-from-scale-in
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So ändern Sie die Instance-Schutzeinstellung für eine Instance

In diesem Beispiel wird der Instance-Schutz für die angegebene Instance deaktiviert.

```
aws autoscaling set-instance-protection \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg \  
  --no-protected-from-scale-in
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [SetInstanceProtection](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Instance-Schutz für die angegebene Instance aktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $true
```

Beispiel 2: In diesem Beispiel wird der Instance-Schutz für die angegebene Instanz deaktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $false
```

- Einzelheiten zur API finden Sie unter [SetInstanceProtection AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Instance-Schutz für die angegebene Instance aktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -ProtectedFromScaleIn $true
```

Beispiel 2: In diesem Beispiel wird der Instance-Schutz für die angegebene Instanz deaktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -ProtectedFromScaleIn $false
```

- Einzelheiten zur API finden Sie unter [SetInstanceProtection AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **SuspendProcesses** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SuspendProcesses` verwendet wird.

CLI

AWS CLI

So setzen Sie Auto-Scaling-Prozesse aus

In diesem Beispiel wird der angegebene Skalierungsprozess für die spezifizierte Auto-Scaling-Gruppe ausgesetzt.

```
aws autoscaling suspend-processes \  
  --auto-scaling-group-name my-asg \  
  --scaling-processes AlarmNotification
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen erhalten Sie unter [Aus- und Fortsetzen von Skalierungsprozessen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie [SuspendProcesses](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess  
"AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [SuspendProcesses AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess  
"AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter [SuspendProcesses AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Verwenden Sie diesen Service mit einem SDK AWS](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **TerminateInstanceInAutoScalingGroup** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TerminateInstanceInAutoScalingGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Terminate all instances in the Auto Scaling group in preparation for
/// deleting the group.
/// </summary>
/// <param name="instanceId">The instance Id of the instance to terminate.</
param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
    string instanceId)
{
    var request = new TerminateInstanceInAutoScalingGroupRequest
    {
        InstanceId = instanceId,
```

```
        ShouldDecrementDesiredCapacity = false,
    };

    var response = await
        _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You have terminated the instance: {instanceId}");
        return true;
    }

    Console.WriteLine($"Could not terminate {instanceId}");
    return false;
}
```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
request.SetInstanceId(instanceIDs[instanceNumber - 1]);
request.SetShouldDecrementDesiredCapacity(false);
```

```
Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
    autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Waiting for EC2 instance with ID '"
            << instanceIDs[instanceNumber - 1] << "' to terminate..."
            << std::endl;
    }
    else {
        std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}
```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So beenden Sie eine Instance in einer Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Instance in der angegebenen Auto-Scaling-Gruppe beendet, ohne dass die Größe der Gruppe aktualisiert wird. Amazon EC2 Auto Scaling startet eine Ersatz-Instance, nachdem die angegebene Instance beendet wurde.

```
aws autoscaling terminate-instance-in-auto-scaling-group \
  --instance-id i-061c63c5eb45f0416 \
  --no-should-decrement-desired-capacity
```

Ausgabe:

```
{
  "Activities": [
    {
      "ActivityId": "8c35d601-793c-400c-fcd0-f64a27530df7",
```

```

        "AutoScalingGroupName": "my-asg",
        "Description": "Terminating EC2 instance: i-061c63c5eb45f0416",
        "Cause": "",
        "StartTime": "2020-10-31T20:34:25.680Z",
        "StatusCode": "InProgress",
        "Progress": 0,
        "Details": "{\"Subnet ID\": \"subnet-6194ea3b\", \"Availability Zone\": \"us-west-2c\"}"
    }
]
}

```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

```
}
```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.


```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

 Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function terminateInstanceInAutoScalingGroup(
    $instanceId,
    $shouldDecrementDesiredCapacity = true,
    $attempts = 0
) {
    try {
        return $this->autoScalingClient-
>terminateInstanceInAutoScalingGroup([
            'InstanceId' => $instanceId,
            'ShouldDecrementDesiredCapacity' =>
$shouldDecrementDesiredCapacity,
        ]);
    } catch (AutoScalingException $exception) {
        if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
$attempts < 5) {
            error_log("Cannot terminate an instance while it is still
pending. Waiting then trying again.");
            sleep(5 * (1 + $attempts));
            return $this->terminateInstanceInAutoScalingGroup(
                $instanceId,
                $shouldDecrementDesiredCapacity,
                ++$attempts
            );
        } else {
            throw $exception;
        }
    }
}
```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet und die gewünschte Kapazität ihrer Auto-Scaling-Gruppe reduziert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5  
AutoScalingGroupName :  
Cause               : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out  
                    of service in response to a user  
                    request, shrinking the capacity from 2 to 1.  
Description         : Terminating EC2 instance: i-93633f9b  
Details             : {"Availability Zone":"us-west-2b","Subnet  
                    ID":"subnet-5264e837"}  
EndTime             :  
Progress            : 0  
StartTime           : 11/22/2015 8:09:03 AM  
StatusCode          : InProgress  
StatusMessage       :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance beendet, ohne die gewünschte Kapazität ihrer Auto Scaling-Gruppe zu reduzieren. Auto Scaling startet eine neue Ersatz-Instance.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5  
AutoScalingGroupName :  
Cause               : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out  
                    of service in response to a user  
                    request.  
Description         : Terminating EC2 instance: i-93633f9b
```

```

Details           : {"Availability Zone":"us-west-2b","Subnet
  ID":"subnet-5264e837"}
EndTime           :
Progress          : 0
StartTime         : 11/22/2015 8:09:03 AM
StatusCode        : InProgress
StatusMessage     :

```

- Einzelheiten zur API finden Sie unter [TerminateInstanceInAutoScalingGroup AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet und die gewünschte Kapazität ihrer Auto-Scaling-Gruppe reduziert, sodass Auto Scaling keine Ersatz-Instance startet.

```

Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true

```

Ausgabe:

```

ActivityId        : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause             : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
  of service in response to a user
                   request, shrinking the capacity from 2 to 1.
Description       : Terminating EC2 instance: i-93633f9b
Details           : {"Availability Zone":"us-west-2b","Subnet
  ID":"subnet-5264e837"}
EndTime           :
Progress          : 0
StartTime         : 11/22/2015 8:09:03 AM
StatusCode        : InProgress
StatusMessage     :

```

Beispiel 2: In diesem Beispiel wird die angegebene Instance beendet, ohne die gewünschte Kapazität ihrer Auto Scaling-Gruppe zu reduzieren. Auto Scaling startet eine neue Ersatz-Instance.

```

Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false

```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
                      of service in response to a user
                      request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter [TerminateInstancesInAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz \(V5\)](#).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
```

```

) -> Dict[str, Any]:
    """
    Stops an instance.

    :param instance_id: The ID of the instance to stop.
    :param decrease_capacity: Specifies whether to decrease the desired
capacity
of the group. When passing True for this
parameter,
you can stop an instance without having a
replacement
instance start when the desired capacity
threshold is
crossed.
    :return: A dictionary containing details of the scaling activity that
occurs
in response to this action.
    :raises ClientError: If there is an error terminating the instance.
    """
    try:
        response =
self.autoscaling_client.terminate_instance_in_auto_scaling_group(
            InstanceId=instance_id,
ShouldDecrementDesiredCapacity=decrease_capacity
        )
        logger.info(f"Successfully terminated instance {instance_id}.")
        return response["Activity"]

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(f"Failed to terminate instance {instance_id}.")
        if error_code == "ScalingActivityInProgress":
            logger.error(
                "A scaling activity is currently in progress for the Auto
Scaling group "
                f"associated with instance '{instance_id}'. "
                "Please wait for the activity to complete before attempting
to terminate the instance."
            )
        elif error_code == "ResourceInUse":
            logger.error(
                f"The instance '{instance_id}' or an associated resource is
currently in use "
                "and cannot be terminated. "

```

```

        "Ensure the instance is not involved in any ongoing processes
and try again."
    )
    logger.error(f"Full error:\n\t{err}")
    raise

```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
    // Retrieve a list of instances in the auto scaling group.
    let auto_scaling_group = self.get_group().await?;
    let instances = auto_scaling_group.instances();
    // Or use other logic to find an instance to terminate.
    let instance = instances.first();
    if let Some(instance) = instance {
        let instance_id = if let Some(instance_id) = instance.instance_id() {
            instance_id
        } else {
            return Err(ScenarioError::with("Missing instance id"));
        };
        let termination = self
            .ec2
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await;
        if let Err(err) = termination {
            Err(ScenarioError::new(

```

```
        "There was a problem terminating an instance",
        &err,
    ))
    } else {
        Ok(())
    }
} else {
    Err(ScenarioError::with("There was no instance to terminate"))
}
}

async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
    let describe_auto_scaling_groups = self
        .autoscaling
        .describe_auto_scaling_groups()
        .auto_scaling_group_names(self.auto_scaling_group_name.clone())
        .send()
        .await;

    if let Err(err) = describe_auto_scaling_groups {
        return Err(ScenarioError::new(
            format!(
                "Failed to get status of autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )
            .as_str(),
            &err,
        ));
    }

    let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
    let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
    let auto_scaling_group = auto_scaling_groups.first();

    if auto_scaling_group.is_none() {
        return Err(ScenarioError::with(format!(
            "Could not find autoscaling group {}",
            self.auto_scaling_group_name.clone()
        )));
    }

    Ok(auto_scaling_group.unwrap().clone())
}
```

```
}
```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel](#) einrichten und ausführen.

```
" Example: iv_instance_id = 'i-1234567890abcdef0'
" Example: iv_decrease_capacity = abap_true

TRY.
  DATA(lo_output) = ao_asc->terminateinstinautoscgrou(
    iv_instanceid = iv_instance_id
    iv_shoulddecrementdesiredcap = iv_decrease_capacity ).

  oo_output = lo_output->get_activity( ).

  MESSAGE 'Instance terminated successfully' TYPE 'I'.

CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
  RAISE EXCEPTION lo_activity_in_progress.
CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
  RAISE EXCEPTION lo_contention.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
  RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **UpdateAutoScalingGroup** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie UpdateAutoScalingGroup verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Update the capacity of an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
/// <param name="maxSize">The maximum number of instances that can be
/// created for the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> UpdateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    int maxSize)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
```

```
};

var groupRequest = new UpdateAutoScalingGroupRequest
{
    MaxSize = maxSize,
    AutoScalingGroupName = groupName,
    LaunchTemplate = templateSpecification,
};

var response = await
_amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
    return true;
}
else
{
    return false;
}
}
```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";
```

```
Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);
request.SetMaxSize(3);

Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
    autoScalingClient.UpdateAutoScalingGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
}
}
```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So aktualisieren Sie die Größenbeschränkungen einer Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe mit einer Mindestgröße von 2 und einer Höchstgröße von 10 aktualisiert.

```
aws autoscaling update-auto-scaling-group \
  --auto-scaling-group-name my-asg \
  --min-size 2 \
  --max-size 10
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Festlegen der Größenbeschränkungen für Ihre Auto-Scaling-Gruppe](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So fügen Sie Zustandsprüfungen für Elastic Load Balancing hinzu und legen fest, welche Availability Zones und Subnetze verwendet werden sollen

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um Zustandsprüfungen für Elastic Load Balancing hinzuzufügen. Dieser Befehl aktualisiert den Wert von `--vpc-zone-identifier` mit einer Liste von Subnetzen IDs in mehreren Availability Zones.

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --health-check-type ELB \  
  --health-check-grace-period 600 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Elastic Load Balancing und Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So aktualisieren Sie die Platzierungsgruppe und die Beendigungsrichtlinie

In diesem Beispiel werden die zu verwendende Platzierungsgruppe und die Beendigungsrichtlinie aktualisiert.

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --placement-group my-placement-group \  
  --termination-policies "OldestInstance"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 4: So verwenden Sie die neueste Version der Startvorlage

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um die neueste Version der angegebenen Startvorlage zu verwenden.

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Startvorlagen](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 5: So verwenden Sie eine bestimmte Version der Startvorlage

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um eine bestimmte Version einer Startvorlage anstelle der neuesten oder der Standardversion zu verwenden.

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Startvorlagen](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 6: So definieren Sie eine Richtlinie für gemischte Instances und aktivieren den Kapazitätsausgleich

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um eine Richtlinie für gemischte Instances zu verwenden, gleichzeitig wird die Kapazitätsanpassung aktiviert. Mit dieser Struktur können Sie Gruppen mit Spot- und On-Demand-Kapazitäten angeben und verschiedene Startvorlagen für unterschiedliche Architekturen verwenden.

```
aws autoscaling update-auto-scaling-group \  
  --cli-input-json file://~/config.json
```

Inhalt von `config.json`:

```
{  
  "AutoScalingGroupName": "my-asg",  
  "CapacityRebalance": true,  
  "MixedInstancesPolicy": {  
    "LaunchTemplate": {  
      "LaunchTemplateSpecification": {  
        "LaunchTemplateName": "my-launch-template-for-x86",  
        "Version": "$Latest"  
      },  
    },  
  },  
}
```

```
    "Overrides": [
      {
        "InstanceType": "c6g.large",
        "LaunchTemplateSpecification": {
          "LaunchTemplateName": "my-launch-template-for-arm",
          "Version": "$Latest"
        }
      },
      {
        "InstanceType": "c5.large"
      },
      {
        "InstanceType": "c5a.large"
      }
    ],
    "InstancesDistribution": {
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "capacity-optimized"
    }
  }
}
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie unter [UpdateAutoScalingGroup AWS CLI Befehlsreferenz](#).

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel](#) einrichten und ausführen.

```
public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
```

```
        String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
            .waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group
" + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group $groupName")
    }
}
```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function updateAutoScalingGroup($autoScalingGroupName, $args)
{
    if (array_key_exists('MaxSize', $args)) {
        $maxSize = ['MaxSize' => $args['MaxSize']];
    } else {
        $maxSize = [];
    }
    if (array_key_exists('MinSize', $args)) {
        $minSize = ['MinSize' => $args['MinSize']];
    } else {
        $minSize = [];
    }
    $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
    $parameters = array_merge($parameters, $minSize, $maxSize);
    return $this->autoScalingClient->updateAutoScalingGroup($parameters);
}
```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die minimale und maximale Größe der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Beispiel 2: In diesem Beispiel wird die standardmäßige Ruhephase der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Beispiel 3: In diesem Beispiel werden die Availability Zones der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-west-2a", "us-west-2b")
```

Beispiel 4: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um Zustandsprüfungen mit Elastic Load Balancing zu verwenden.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -  
HealthCheckGracePeriod 60
```

- Einzelheiten zur API finden Sie unter [UpdateAutoScalingGroup AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die minimale und maximale Größe der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Beispiel 2: In diesem Beispiel wird die standardmäßige Ruhephase der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Beispiel 3: In diesem Beispiel werden die Availability Zones der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-west-2a", "us-west-2b")
```

Beispiel 4: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um Zustandsprüfungen mit Elastic Load Balancing zu verwenden.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Einzelheiten zur API finden Sie unter [UpdateAutoScalingGroup AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def update_group(self, group_name: str, **kwargs: Any) -> None:
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        :return: None
        :raises ClientError: If there is an error updating the Auto Scaling
        group.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
```

```

        AutoScalingGroupName=group_name, **kwargs
    )
    logger.info(f"Successfully updated Auto Scaling group {group_name}.")

except ClientError as err:
    error_code = err.response["Error"]["Code"]
    logger.error(f"Failed to update Auto Scaling group {group_name}.")
    if error_code == "ResourceInUse":
        logger.error(
            "The Auto Scaling group '%s' is currently in use and cannot
be modified. Please try again later.",
            group_name,
        )
    elif error_code == "ScalingActivityInProgress":
        logger.error(
            f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'."
            "Please wait for the activity to complete before attempting
to update the group."
        )
    logger.error(f"Full error:\n\t{err}")
    raise

```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

async fn update_group(client: &Client, name: &str, size: i32) -> Result<(),
Error> {
    client

```

```

        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
        .send()
        .await?;

println!("Updated AutoScaling group");

Ok(())
}

```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_max_size = 5

TRY.
  ao_asc->updateautoscalinggroup(
    iv_autoscalinggroupname = iv_group_name
    iv_maxsize = iv_max_size
    iv_minsize = iv_min_size ).

  MESSAGE 'Auto Scaling group updated successfully' TYPE 'I'.

CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
  RAISE EXCEPTION lo_contention.
CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
  RAISE EXCEPTION lo_activity_in_progress.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).

```

```
RAISE EXCEPTION lo_generic_exception.  
ENDTRY.
```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Auto Scaling mit implementieren AWS SDKs. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Auto Scaling oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- [Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK](#)

Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Webservice mit Lastenausgleich erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.

- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webserver auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
static async Task Main(string[] args)
{
    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
        .Build();

    // Set up dependency injection for the AWS services.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft",
                    LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft",
                    LogLevel.Trace))
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>())
```

```
        .AddAWSService<IAmazonDynamoDB>()
        .AddAWSService<IAmazonElasticLoadBalancingV2>()
        .AddAWSService<IAmazonSimpleSystemsManagement>()
        .AddAWSService<IAmazonAutoScaling>()
        .AddAWSService<IAmazonEC2>()
        .AddTransient<AutoScalerWrapper>()
        .AddTransient<ElasticLoadBalancerWrapper>()
        .AddTransient<SmParameterWrapper>()
        .AddTransient<Recommendations>()
        .AddSingleton<IConfiguration>(_configuration)
    )
    .Build();

    ServicesSetup(host);
    ResourcesSetup();

    try
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
        Console.WriteLine(new string('-', 80));
        await Deploy(true);

        Console.WriteLine("Now let's begin the scenario.");
        Console.WriteLine(new string('-', 80));
        await Demo(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Finally, let's clean up our resources.");
        Console.WriteLine(new string('-', 80));

        await DestroyResources(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
    }
}
```

```
        await DestroyResources(true);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Setup any common resources, also used for integration testing.
/// </summary>
public static void ResourcesSetup()
{
    _httpClient = new HttpClient();
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
    _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    _recommendations = host.Services.GetRequiredService<Recommendations>();
    _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
    _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
}

/// <summary>
/// Deploy necessary resources for the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Deploy(bool interactive)
{
    var protocol = "HTTP";
    var port = 80;
    var sshPort = 22;

    Console.WriteLine(
        "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
```

```
        "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
        "against various kinds of failures.\n\n" +
        "Some of the resources create by this demo are:\n");

    Console.WriteLine(
        "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
    Console.WriteLine(
        "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
    Console.WriteLine(
        "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
    Console.WriteLine(
        "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
    if (interactive)
        Console.ReadLine();

    // Create and populate the DynamoDB table.
    var databaseTableName = _configuration["databaseName"];
    var recommendationsPath = Path.Join(_configuration["resourcePath"],
        "recommendations_objects.json");
    Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
    await _recommendations.CreateDatabaseWithName(databaseTableName);
    await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
    Console.WriteLine(new string('-', 80));

    // Create the EC2 Launch Template.

    Console.WriteLine(
        $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
        + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
        + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
```

```
        + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
        + "run a web server, such as Apache, with least-privileged
credentials.");
    Console.WriteLine(
        "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
        + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
        + "that control the flow of the demo.");

    var startupScriptPath = Path.Join(_configuration["resourcePath"],
        "server_startup_script.sh");
    var instancePolicyPath = Path.Join(_configuration["resourcePath"],
        "instance_policy.json");
    await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
    Console.WriteLine(new string('-', 80));

    Console.WriteLine(
        "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
        + "Availability Zone.\n");
    var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
    await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
    Console.WriteLine(new string('-', 80));

    Console.WriteLine(
        "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
        + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you're ready to continue.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("Creating variables that control the flow of the
demo.");
    await _smParameterWrapper.Reset();

    Console.WriteLine(
```

```
        "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
        + "defines how the load balancer connects to instances. The load
balancer provides a\n"
        + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
_elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroup
protocol, port, defaultVpc.VpcId);

        await
_elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
subnetIds, targetGroup);
        await
_autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
_elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
var loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
_autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
_autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
ipString);
            var sshPortIsOpen =
_autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
ipString);
```

```
        if (!portIsOpen)
        {
            Console.WriteLine(
                "\nFor this example to work, the default security group for
your default VPC must\n"
                + "allows access from this computer. You can either add it
automatically from this\n"
                + "example or add it yourself using the AWS Management
Console.\n");

            if (!interactive || GetYesNoResponse(
                "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
            {
                await
                _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
                ipString);
            }
        }

        if (!sshPortIsOpen)
        {
            if (!interactive || GetYesNoResponse(
                "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
            {
                await
                _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
                ipString);
            }
            loadBalancerAccess = await
            _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"http://{endPoint}\n");
        }
        else
        {
```

```
        Console.WriteLine(
            "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
            + "manually verifying that your VPC and security group are
configured correctly and that\n"
            + "you can successfully make a GET request to the load balancer
endpoint:\n");
        Console.WriteLine($"http://{endPoint}\n");
    }
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
    if (interactive)
        Console.ReadLine();
    return true;
}

/// <summary>
/// Demonstrate the steps of the scenario.
/// </summary>
/// <param name="interactive">True to run as an interactive scenario.</param>
/// <returns>Async task.</returns>
public static async Task<bool> Demo(bool interactive)
{
    var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
        "ssm_only_policy.json");

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Resetting parameters to starting values for demo.");
    await _smParameterWrapper.Reset();

    Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
        "to create situations where the web service fails, and
shows how using a resilient\n" +
        "architecture can keep the web service running in spite
of these failures.");
    Console.WriteLine(new string('-', 88));
    Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
    if (interactive)
        await DemoActionChoices();
}
```

```
    Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
        $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
        $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
    await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
    "this-is-not-a-table");
    Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
        "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
    if (interactive)
        await DemoActionChoices();

    Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
    Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

    await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
    "static");

    Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
    Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
    if (interactive)
        await DemoActionChoices();

    Console.WriteLine("Let's reinstate the recommendation service.\n");
    await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
    _smParameterWrapper.TableName);
    Console.WriteLine(
        "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
        "access the DynamoDB recommendation table.\n"
    );
    await _autoScalerWrapper.CreateInstanceProfileWithName(
        _autoScalerWrapper.BadCredsPolicyName,
        _autoScalerWrapper.BadCredsRoleName,
```

```
        _autoScalerWrapper.BadCredsProfileName,
        ssmOnlyPolicy,
        new List<string> { "AmazonSSMManagedInstanceCore" }
    );
    var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
    var badInstanceId = instances.First();
    var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
    Console.WriteLine(
        $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
        "bad credentials...\n"
    );
    await _autoScalerWrapper.ReplaceInstanceProfile(
        badInstanceId,
        _autoScalerWrapper.BadCredsProfileName,
        instanceProfile.AssociationId
    );
    Console.WriteLine(
        "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
        "depending on which instance is selected by the load balancer.\n"
    );
    if (interactive)
        await DemoActionChoices();

    Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
    Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
    Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
    Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
    Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

    Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
    Console.WriteLine("and take that instance out of rotation.");
```

```
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
```

```

        Console.WriteLine($"\\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
                _elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
                _elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
                _autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
                _autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
                _autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName

```

```
        );
        await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
    }
    else
    {
        Console.WriteLine(
            "Ok, we'll leave the resources intact.\n" +
            "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
        );
    }

    Console.WriteLine(new string('-', 80));
    return true;
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";
```

```
public string GroupName => _groupName;
public string KeyPairName => _keyPairName;
public string LaunchTemplateName => _launchTemplateName;
public string InstancePolicyName => _instancePolicyName;
public string BadCredsProfileName => _badCredsProfileName;
public string BadCredsRoleName => _badCredsRoleName;
public string BadCredsPolicyName => _badCredsPolicyName;

/// <summary>
/// Constructor for the AutoScalerWrapper.
/// </summary>
/// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
/// <param name="amazonEc2">The injected EC2 client.</param>
/// <param name="amazonIam">The injected IAM client.</param>
/// <param name="amazonSsm">The injected SSM client.</param>
public AutoScalerWrapper(
    IAmazonAutoScaling amazonAutoScaling,
    IAmazonEC2 amazonEc2,
    IAmazonSimpleSystemsManagement amazonSsm,
    IAmazonIdentityManagementService amazonIam,
    IConfiguration configuration,
    ILogger<AutoScalerWrapper> logger)
{
    _amazonAutoScaling = amazonAutoScaling;
    _amazonEc2 = amazonEc2;
    _amazonSsm = amazonSsm;
    _amazonIam = amazonIam;
    _logger = logger;

    var prefix = configuration["resourcePrefix"];
    _instanceType = configuration["instanceType"];
    _amiParam = configuration["amiParam"];

    _launchTemplateName = prefix + "-template";
    _groupName = prefix + "-group";
    _instancePolicyName = prefix + "-pol";
    _instanceRoleName = prefix + "-role";
    _instanceProfileName = prefix + "-prof";
    _badCredsPolicyName = prefix + "-bc-pol";
    _badCredsRoleName = prefix + "-bc-role";
    _badCredsProfileName = prefix + "-bc-prof";
    _keyPairName = prefix + "-key-pair";
}
```

```

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
    a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance. The role has attached policies that specify the AWS permissions
    granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
    the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
            "\"Version\": \"2012-10-17\", " +
            "\"Statement\": [{" +
                "\"Effect\": \"Allow\", " +
                "\"Principal\": {" +
                "\"Service\": [" +
                    "\"ec2.amazonaws.com\"" +
                "]" +
                "}, " +
                "\"Action\": \"sts:AssumeRole\"" +
            "]" +
            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest

```

```
        {
            PolicyName = policyName,
            PolicyDocument = policyDocument
        });
        policyArn = createPolicyResult.Policy.Arn;
    }
    catch (EntityAlreadyExistsException)
    {
        // The policy already exists, so we look it up to get the Arn.
        var policiesPaginator = _amazonIam.Paginators.ListPolicies(
            new ListPoliciesRequest()
            {
                Scope = PolicyScopeType.Local
            });
        // Get the entire list using the paginator.
        await foreach (var policy in policiesPaginator.Policies)
        {
            if (policy.PolicyName.Equals(policyName))
            {
                policyArn = policy.Arn;
            }
        }

        if (policyArn == null)
        {
            throw new InvalidOperationException("Policy not found");
        }
    }

    try
    {
        await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = assumeRoleDoc,
        });
        await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
        {
            RoleName = roleName,
            PolicyArn = policyArn
        });
        if (awsManagedPolicies != null)
        {
            foreach (var awsPolicy in awsManagedPolicies)
```

```
        {
            await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
            {
                PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                RoleName = roleName
            });
        }
    }
}
catch (EntityAlreadyExistsException)
{
    Console.WriteLine("Role already exists.");
}

string profileArn = "";
try
{
    var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
        new CreateInstanceProfileRequest()
        {
            InstanceProfileName = profileName
        });
    // Allow time for the profile to be ready.
    profileArn = profileCreateResponse.InstanceProfile.Arn;
    Thread.Sleep(10000);
    await _amazonIam.AddRoleToInstanceProfileAsync(
        new AddRoleToInstanceProfileRequest()
        {
            InstanceProfileName = profileName,
            RoleName = roleName
        });
}
catch (EntityAlreadyExistsException)
{
    Console.WriteLine("Policy already exists.");
    var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
        new GetInstanceProfileRequest()
        {
            InstanceProfileName = profileName
        });
    profileArn = profileGetResponse.InstanceProfile.Arn;
```

```
    }
    return profileArn;
}

/// <summary>
/// Create a new key pair and save the file.
/// </summary>
/// <param name="newKeyName">The name of the new key pair.</param>
/// <returns>Async task.</returns>
public async Task CreateKeyPair(string newKeyName)
{
    try
    {
        var keyResponse = await _amazonEc2.CreateKeyPairAsync(
            new CreateKeyPairRequest() { KeyName = newKeyName });
        await File.WriteAllTextAsync($"{newKeyName}.pem",
            keyResponse.KeyPair.KeyMaterial);
        Console.WriteLine($"Created key pair {newKeyName}.");
    }
    catch (AlreadyExistsException)
    {
        Console.WriteLine("Key pair already exists.");
    }
}

/// <summary>
/// Delete the key pair and file by name.
/// </summary>
/// <param name="deleteKeyName">The key pair to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteKeyPairByName(string deleteKeyName)
{
    try
    {
        await _amazonEc2.DeleteKeyPairAsync(
            new DeleteKeyPairRequest() { KeyName = deleteKeyName });
        File.Delete($"{deleteKeyName}.pem");
    }
    catch (FileNotFoundException)
    {
        Console.WriteLine($"Key pair {deleteKeyName} not found.");
    }
}
```

```

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
    Scaling.
    /// The launch template specifies a Bash script in its user data field that
    runs after
    /// the instance is started. This script installs the Python packages and
    starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
    run.</param>
    /// <param name="instancePolicyPath">The path to a permissions policy to
    create and attach to the profile.</param>
    /// <returns>The template object.</returns>
    public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
    startupScriptPath, string instancePolicyPath)
    {
        try
        {
            await CreateKeyPair(_keyPairName);
            await CreateInstanceProfileWithName(_instancePolicyName,
            _instanceRoleName,
                _instanceProfileName, instancePolicyPath);

            var startServerText = await File.ReadAllTextAsync(startupScriptPath);
            var plainTextBytes =
            System.Text.Encoding.UTF8.GetBytes(startServerText);

            var amiLatest = await _amazonSsm.GetParameterAsync(
                new GetParameterRequest() { Name = _amiParam });
            var amiId = amiLatest.Parameter.Value;
            var launchTemplateResponse = await
            _amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new
            LaunchTemplateIamInstanceProfileSpecificationRequest()

```

```

        {
            Name = _instanceProfileName
        },
        KeyName = _keyPairName,
        UserData = System.Convert.ToBase64String(plainTextBytes)
    }
    });
    return launchTemplateResponse.LaunchTemplate;
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode ==
        "InvalidLaunchTemplateName.AlreadyExistsException")
    {
        _logger.LogError($"Could not create the template, the name
        {_launchTemplateName} already exists. " +
            $"Please try again with a unique name.");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError($"An error occurred while creating the template.:
    {ex.Message}");
    throw;
}
}

/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    try
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z =>
            z.ZoneName).ToList();
    }
    catch (AmazonEC2Exception ec2Exception)

```

```
    {
        _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Create an EC2 Auto Scaling group of a specified size and name.
/// </summary>
/// <param name="groupSize">The size for the group.</param>
/// <param name="groupName">The name for the group.</param>
/// <param name="availabilityZones">The availability zones for the group.</
param>
/// <returns>Async task.</returns>
public async Task CreateGroupOfSize(int groupSize, string groupName,
List<string> availabilityZones)
{
    try
    {
        await _amazonAutoScaling.CreateAutoScalingGroupAsync(
            new CreateAutoScalingGroupRequest()
            {
                AutoScalingGroupName = groupName,
                AvailabilityZones = availabilityZones,
                LaunchTemplate =
                    new
Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                    {
                        LaunchTemplateName = _launchTemplateName,
                        Version = "$Default"
                    },
                MaxSize = groupSize,
                MinSize = groupSize
            });
        Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
size {groupSize}.");
    }
}
```

```
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
```

```
/// Get all the subnets for a Vpc in a set of availability zones.
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    try
    {
        var subnets = new List<Subnet>();
        var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
            new DescribeSubnetsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("vpc-id", new List<string>() { vpcId }),
                    new("availability-zone", availabilityZones),
                    new("default-for-az", new List<string>() { "true" })
                }
            });

        // Get the entire list using the paginator.
        await foreach (var subnet in subnetPaginator.Subnets)
        {
            subnets.Add(subnet);
        }

        return subnets;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
        {
            _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
    }
}
```

```
        throw;
    }
}

/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
                LaunchTemplateName = templateName
            });
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
            "InvalidLaunchTemplateName.NotFoundException")
        {
            _logger.LogError(
                $"Could not delete the template, the name
                {_launchTemplateName} was not found.");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while deleting the template.:
        {ex.Message}");
        throw;
    }
}

/// <summary>
/// Detaches a role from an instance profile, detaches policies from the
role,
/// and deletes all the resources.
/// </summary>
```

```
/// <param name="profileName">The name of the profile to delete.</param>
/// <param name="roleName">The name of the role to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteInstanceProfile(string profileName, string roleName)
{
    try
    {
        await _amazonIam.RemoveRoleFromInstanceProfileAsync(
            new RemoveRoleFromInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
                RoleName = roleName
            });
        await _amazonIam.DeleteInstanceProfileAsync(
            new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
        var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
            new ListAttachedRolePoliciesRequest() { RoleName = roleName });
        foreach (var policy in attachedPolicies.AttachedPolicies)
        {
            await _amazonIam.DetachRolePolicyAsync(
                new DetachRolePolicyRequest()
                {
                    RoleName = roleName,
                    PolicyArn = policy.PolicyArn
                });
            // Delete the custom policies only.
            if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
            {
                await _amazonIam.DeletePolicyAsync(
                    new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                    {
                        PolicyArn = policy.PolicyArn
                    });
            }
        }

        await _amazonIam.DeleteRoleAsync(
            new DeleteRoleRequest() { RoleName = roleName });
    }
    catch (NoSuchEntityException)
    {
        Console.WriteLine($"Instance profile {profileName} does not exist.");
    }
}
```

```
    }
}

/// <summary>
/// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
/// </summary>
/// <param name="group">The name of the auto scaling group.</param>
/// <returns>A collection of instance Ids.</returns>
public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
{
    var instanceResponse = await
    _amazonAutoScaling.DescribeAutoScalingGroupsAsync(
        new DescribeAutoScalingGroupsRequest()
        {
            AutoScalingGroupNames = new List<string>() { group }
        });
    var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
        g => g.Instances.Select(i => i.InstanceId));
    return instanceIds;
}

/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    try
    {
        var response = await
    _amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
        new DescribeIamInstanceProfileAssociationsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new("instance-id", new List<string>() { instanceId })
            },
        });
        return response.IamInstanceProfileAssociations[0];
    }
    catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                }
            );
        }
    }
}
```

```

// Allow time before resetting.
Thread.Sleep(25000);

await _amazonEc2.RebootInstancesAsync(
    new RebootInstancesRequest(new List<string>() { instanceId }));
Thread.Sleep(25000);
var instanceReady = false;
var retries = 5;
while (retries-- > 0 && !instanceReady)
{
    var instancesPaginator =
        _amazonSsm.Paginators.DescribeInstanceInformation(
            new DescribeInstanceInformationRequest());
    // Get the entire list using the paginator.
    await foreach (var instance in
instancesPaginator.InstanceInformationList)
    {
        instanceReady = instance.InstanceId == instanceId;
        if (instanceReady)
        {
            break;
        }
    }
}
Console.WriteLine("Waiting for instance to be running.");
await WaitForInstanceState(instanceId, InstanceStateName.Running);
Console.WriteLine("Instance ready.");
Console.WriteLine($"Sending restart command to instance
{instanceId}");
await _amazonSsm.SendCommandAsync(
    new SendCommandRequest()
    {
        InstanceIds = new List<string>() { instanceId },
        DocumentName = "AWS-RunShellScript",
        Parameters = new Dictionary<string, List<string>>()
        {
            {
                "commands",
                new List<string>() { "cd / && sudo python3 server.py
80" }
            }
        }
    }
});

```

```
        Console.WriteLine($"Restarted the web server on instance
{instanceId}");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Try to terminate an instance by its Id.
/// </summary>
/// <param name="instanceId">The Id of the instance to terminate.</param>
/// <returns>Async task.</returns>
public async Task TryTerminateInstanceById(string instanceId)
{
    var stopping = false;
    Console.WriteLine($"Stopping {instanceId}...");
    while (!stopping)
    {
        try
        {
            await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                new TerminateInstanceInAutoScalingGroupRequest()
                {
                    InstanceId = instanceId,
                    ShouldDecrementDesiredCapacity = false
                });
            stopping = true;
        }
        catch (ScalingActivityInProgressException)
```

```
        {
            Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
/// waits and retries until the group is successfully deleted.
/// </summary>
/// <param name="groupName">The name of the group to try to delete.</param>
/// <returns>Async task.</returns>
public async Task TryDeleteGroupByName(string groupName)
{
    var stopped = false;
    while (!stopped)
    {
        try
        {
            await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                new DeleteAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName
                });
            stopped = true;
        }
        catch (Exception e)
            when ((e is ScalingActivityInProgressException)
                || (e is Amazon.AutoScaling.Model.ResourceInUseException))
        {
            Console.WriteLine($"Some instances are still running.
Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Terminate instances and delete the Auto Scaling group by name.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
```

```

    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }

    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {

```

```

        Filters = new List<Amazon.EC2.Model.Filter>()
        {
            new ("group-name", new List<string>() { "default" }),
            new ("vpc-id", new List<string>() { vpc.VpcId })
        }
    });
    return groupResponse.SecurityGroups[0];
}

/// <summary>
/// Verify the default security group of a Vpc allows ingress from the
calling computer.
/// This can be done by allowing ingress from this computer's IP address.
/// In some situations, such as connecting from a corporate network, you must
instead specify
/// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
/// If you do, be sure to remove public access when you're done.
/// </summary>
/// <param name="vpc">The group to check.</param>
/// <param name="port">The port to verify.</param>
/// <param name="ipAddress">This computer's IP address.</param>
/// <returns>True if the ip address is allowed on the group.</returns>
public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
{
    var portIsOpen = false;
    foreach (var ipPermission in group.IpPermissions)
    {
        if (ipPermission.FromPort == port)
        {
            foreach (var ipRange in ipPermission.Ipv4Ranges)
            {
                var cidr = ipRange.CidrIp;
                if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                {
                    portIsOpen = true;
                }
            }

            if (ipPermission.PrefixListIds.Any())
            {
                portIsOpen = true;
            }
        }
    }
}

```

```
        if (!portIsOpen)
        {
            Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
        }
        else
        {
            break;
        }
    }
}

return portIsOpen;
}

/// <summary>
/// Add an ingress rule to the specified security group that allows access on
the
/// specified port from the specified IP address.
/// </summary>
/// <param name="groupId">The Id of the security group to modify.</param>
/// <param name="port">The port to open.</param>
/// <param name="ipAddress">The IP address to allow access.</param>
/// <returns>Async task.</returns>
public async Task OpenInboundPort(string groupId, int port, string ipAddress)
{
    await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
        new AuthorizeSecurityGroupIngressRequest()
        {
            GroupId = groupId,
            IpPermissions = new List<IpPermission>()
            {
                new IpPermission()
                {
                    FromPort = port,
                    ToPort = port,
                    IpProtocol = "tcp",
                    Ipv4Ranges = new List<IpRange>()
                    {
                        new IpRange() { CidrIp = $"{ipAddress}/32" }
                    }
                }
            }
        }
    );
}
```

```

        }
    }
    });
}

/// <summary>
/// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
/// The
/// </summary>
/// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
/// <param name="targetGroupArn">The Arn for the target group.</param>
/// <returns>Async task.</returns>
public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
string targetGroupArn)
{
    await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
        new AttachLoadBalancerTargetGroupsRequest()
        {
            AutoScalingGroupName = autoScalingGroupName,
            TargetGroupARNs = new List<string>() { targetGroupArn }
        });
}

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

    // Wait until the instance is in the specified state.
    var hasState = false;
    do
    {
        // Wait 5 seconds.

```

```

        Thread.Sleep(5000);

        // Check for the desired state.
        var response = await _amazonEc2.DescribeInstancesAsync(request);
        var instance = response.Reservations[0].Instances[0];
        hasState = instance.State.Name == stateName;
        Console.WriteLine(". ");
    } while (!hasState);

    return hasState;
}
}

```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```

/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
    client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
    }
}

```

```
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
```

```
public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
{
    List<TargetHealthDescription> result = null!;
    try
    {
        var groupResponse =
            await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                new DescribeTargetGroupsRequest()
                {
                    Names = new List<string>() { groupName }
                });
        var healthResponse =
            await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                new DescribeTargetHealthRequest()
                {
                    TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                });
        ;
        result = healthResponse.TargetHealthDescriptions;
    }
    catch (TargetGroupNotFoundException)
    {
        Console.WriteLine($"Target group {groupName} not found.");
    }
    return result;
}

/// <summary>
/// Create an Elastic Load Balancing target group. The target group specifies
how the load balancer forwards
/// requests to instances in the group and how instance health is checked.
///
/// To speed up this demo, the health check is configured with shortened
times and lower thresholds. In production,
/// you might want to decrease the sensitivity of your health checks to avoid
unwanted failures.
/// </summary>
/// <param name="groupName">The name for the group.</param>
/// <param name="protocol">The protocol, such as HTTP.</param>
/// <param name="port">The port to use to forward requests, such as 80.</
param>
```

```
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
        _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
        _amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;
```

```
// Wait for load balancer to be available.
var loadBalancerReady = false;
while (!loadBalancerReady)
{
    try
    {
        var describeResponse =
            await
            _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                new DescribeLoadBalancersRequest()
                {
                    Names = new List<string>() { name }
                });

        var loadBalancerState =
            describeResponse.LoadBalancers[0].State.Code;

        loadBalancerReady = loadBalancerState ==
            LoadBalancerStateEnum.Active;
    }
    catch (LoadBalancerNotFoundException)
    {
        loadBalancerReady = false;
    }
    Thread.Sleep(10000);
}
// Create the listener.
await _amazonElasticLoadBalancingV2.CreateListenerAsync(
    new CreateListenerRequest()
    {
        LoadBalancerArn = loadBalancerArn,
        Protocol = targetGroup.Protocol,
        Port = targetGroup.Port,
        DefaultActions = new List<Action>()
        {
            new Action()
            {
                Type = ActionTypeEnum.Forward,
                TargetGroupArn = targetGroup.TargetGroupArn
            }
        }
    });
return createLbResponse.LoadBalancers[0];
```

```
}

/// <summary>
/// Verify this computer can successfully send a GET request to the
/// load balancer endpoint.
/// </summary>
/// <param name="endpoint">The endpoint to check.</param>
/// <returns>True if successful.</returns>
public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
{
    var success = false;
    var retries = 3;
    while (!success && retries > 0)
    {
        try
        {
            var endpointResponse = await _httpClient.GetAsync($"http://{
{endpoint}");
            Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

            if (endpointResponse.IsSuccessStatusCode)
            {
                success = true;
            }
            else
            {
                retries = 0;
            }
        }
        catch (HttpRequestException)
        {
            Console.WriteLine("Connection error, retrying...");
            retries--;
            Thread.Sleep(10000);
        }
    }

    return success;
}

/// <summary>
/// Delete a load balancer by its specified name.
/// </summary>
/// <param name="name">The name of the load balancer to delete.</param>
```

```
/// <returns>Async task.</returns>
public async Task DeleteLoadBalancerByName(string name)
{
    try
    {
        var describeLoadBalancerResponse =
            await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                new DescribeLoadBalancersRequest()
                {
                    Names = new List<string>() { name }
                });
        var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
        await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
            new DeleteLoadBalancerRequest()
            {
                LoadBalancerArn = lbArn
            }
        );
    }
    catch (LoadBalancerNotFoundException)
    {
        Console.WriteLine($"Load balancer {name} not found.");
    }
}

/// <summary>
/// Delete a TargetGroup by its specified name.
/// </summary>
/// <param name="groupName">Name of the group to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTargetGroupByName(string groupName)
{
    var done = false;
    while (!done)
    {
        try
        {
            var groupResponse =
                await
amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    }
                );
        }
        catch { }
    }
}
```



```
/// </summary>
/// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
/// <param name="configuration">The injected configuration.</param>
public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
{
    _amazonDynamoDb = amazonDynamoDb;
    _context = new DynamoDBContext(_amazonDynamoDb);
    _tableName = configuration["databaseName"]!;
}

/// <summary>
/// Create the DynamoDb table with a specified name.
/// </summary>
/// <param name="tableName">The name for the table.</param>
/// <returns>True when ready.</returns>
public async Task<bool> CreateDatabaseWithName(string tableName)
{
    try
    {
        Console.WriteLine($"Creating table {tableName}...");
        var createRequest = new CreateTableRequest()
        {
            TableName = tableName,
            AttributeDefinitions = new List<AttributeDefinition>()
            {
                new AttributeDefinition()
                {
                    AttributeName = "MediaType",
                    AttributeType = ScalarAttributeType.S
                },
                new AttributeDefinition()
                {
                    AttributeName = "ItemId",
                    AttributeType = ScalarAttributeType.N
                }
            },
            KeySchema = new List<KeySchemaElement>()
            {
                new KeySchemaElement()
                {
                    AttributeName = "MediaType",
                    KeyType = KeyType.HASH
                },
            },
        };
    }
}
```

```
        new KeySchemaElement()
        {
            AttributeName = "ItemId",
            KeyType = KeyType.RANGE
        }
    },
    ProvisionedThroughput = new ProvisionedThroughput()
    {
        ReadCapacityUnits = 5,
        WriteCapacityUnits = 5
    }
};
await _amazonDynamoDb.CreateTableAsync(createRequest);

// Wait until the table is ACTIVE and then report success.
Console.WriteLine("\nWaiting for table to become active...");

var request = new DescribeTableRequest
{
    TableName = tableName
};

TableStatus status;
do
{
    Thread.Sleep(2000);

    var describeTableResponse = await
        _amazonDynamoDb.DescribeTableAsync(request);
    status = describeTableResponse.Table.TableStatus;

    Console.WriteLine(".");
}
while (status != "ACTIVE");

return status == TableStatus.ACTIVE;
}
catch (ResourceInUseException)
{
    Console.WriteLine($"Table {tableName} already exists.");
    return false;
}
}
```

```
    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
recommendationsPath)
    {
        var recommendationsText = await
File.ReadAllTextAsync(recommendationsPath);
        var records =

JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
/// parameters
/// to drive the demonstration of resilient architecture, such as failure of a
/// dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
        _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
```

```
/// Reset the Systems Manager parameters to starting values for the demo.
/// </summary>
/// <returns>Async task.</returns>
public async Task Reset()
{
    await this.PutParameterByName(_tableParameter, _tableName);
    await this.PutParameterByName(_failureResponseParameter, "none");
    await this.PutParameterByName(_healthCheckParameter, "shallow");
}

/// <summary>
/// Set the value of a named Systems Manager parameter.
/// </summary>
/// <param name="name">The name of the parameter.</param>
/// <param name="value">The value to set.</param>
/// <returns>Async task.</returns>
public async Task PutParameterByName(string name, string value)
{
    await _amazonSimpleSystemsManagement.PutParameterAsync(
        new PutParameterRequest() { Name = name, Value = value, Overwrite =
true });
}
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)

- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {  
  
    public static final String fileName = "C:\\\\AWS\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\\\AWS\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
    public static final String policyFile = "C:\\\\AWS\\resworkflow\\  
\\instance_policy.json"; // Modify file location.  
}
```

```
public static final String ssmJSON = "C:\\\\AWS\\\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
public static final String templateName = "doc-example-resilience-template";
public static final String roleName = "doc-example-resilience-role";
public static final String policyName = "doc-example-resilience-pol";
public static final String profileName = "doc-example-resilience-prof";

public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

public static final String targetGroupName = "doc-example-resilience-tg";
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\\0",
"-");

public static void main(String[] args) throws IOException,
InterruptedException {
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and
Manage a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
}
```

```
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.

    To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
        """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
```

```

        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
    InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""

```

Creating an EC2 launch template that runs '{startup_script}' when an instance starts.

This script starts a Python web server defined in the `server.py` script. The web server

listens to HTTP requests on port 80 and responds to requests to '/' and to '/healthcheck'.

For demo purposes, this server is run as the root user. In production, the best practice is to

run a web server, such as Apache, with least-privileged credentials.

The template also defines an IAM policy that each instance uses to assume a role that grants

permissions to access the DynamoDB recommendation table and Systems Manager parameters

that control the flow of the demo.

```
""");
```

```
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();  
templateCreator.createTemplate(policyFile, policyName, profileName,  
startScript, templateName, roleName);  
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println(  
    "Creating an EC2 Auto Scaling group that maintains three EC2  
instances, each in a different Availability Zone.");
```

```
System.out.println("*** Wait 30 secs for the VPC to be created");  
TimeUnit.SECONDS.sleep(30);
```

```
AutoScaler autoScaler = new AutoScaler();
```

```
String[] zones = autoScaler.createGroup(3, templateName,  
autoScalingGroupName);  
  
System.out.println("""  
    At this point, you have EC2 instances created. Once each instance  
starts, it listens for  
    HTTP requests. You can see these instances in the console or  
continue with the demo.  
    Press Enter when you're ready to continue.  
""");
```

```
System.out.println("""  
    At this point, you have EC2 instances created. Once each instance  
starts, it listens for  
    HTTP requests. You can see these instances in the console or  
continue with the demo.  
    Press Enter when you're ready to continue.  
""");
```

```
System.out.println("""  
    At this point, you have EC2 instances created. Once each instance  
starts, it listens for  
    HTTP requests. You can see these instances in the console or  
continue with the demo.  
    Press Enter when you're ready to continue.  
""");
```

```
System.out.println("""  
    At this point, you have EC2 instances created. Once each instance  
starts, it listens for  
    HTTP requests. You can see these instances in the console or  
continue with the demo.  
    Press Enter when you're ready to continue.  
""");
```

```
System.out.println("""  
    At this point, you have EC2 instances created. Once each instance  
starts, it listens for  
    HTTP requests. You can see these instances in the console or  
continue with the demo.  
    Press Enter when you're ready to continue.  
""");
```

```
""");
```

```
in.nextLine();
```

```
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the
demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load
balancer. The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
```

```
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group
for your default VPC must
                allow access from this computer. You can either add
it automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
```

```
        System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
        System.out.println("you can successfully make a GET request to the
load balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and
shows how using a resilient
                architecture can keep the web service running in spite
of these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                """);
    demoChoices(loadBalancer);

    System.out.println(
        """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                """);
```

```
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    ""
    \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
    healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
    "");
demoChoices(loadBalancer);

System.out.println(
    ""
    Instead of failing when the recommendation service fails,
the web service can return a static response.
    While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
    "");
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns
a static response.
    The service still reports as healthy because health checks are
still shallow.
    """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance
id value.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
```

```
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " +
profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
+ " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """"
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
        the web service can access the DynamoDB table that it depends on
for recommendations. Note that
        the deep health check is only for ELB routing and not for Auto
Scaling instance health.
        This kind of deep health check is not recommended for Auto
Scaling instance health, because it
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");
```

```
System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
    is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
    instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
    the load balancer takes unhealthy instances out of its rotation.
    """);

demoChoices(loadBalancer);

System.out.println(
    """
        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
        instance is to terminate it and let the auto scaler start
a new instance to replace it.
    """);
autoScaler.terminateInstance(badInstanceId);

System.out.println("""
    Even while the instance is terminating and the new instance is
starting, sending a GET
    request to the web service continues to get a successful
recommendation response because
    the load balancer routes requests to the healthy instances. After
the replacement instance
    starts and reports as healthy, it is included in the load
balancing rotation.
    Note that terminating and replacing an instance typically takes
several minutes, during which time you
    can see the changing health check status until the new instance
is running and healthy.
    """);

demoChoices(loadBalancer);
System.out.println(
    "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

demoChoices(loadBalancer);
paramHelper.reset();
```

```
}

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClientClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
```

```

        new
        InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load
balancer targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
health check to update
                Note that it can take a minute or two for the
                after changes are made.
                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {

```

```
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile
is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
```

```
        throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
        .builder()
        .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
        // name.
        .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
        .builder()
        .iamInstanceProfile(iamInstanceProfile)
        .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
        .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
            newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
        }
    }
}
```

```
        try {
            TimeUnit.SECONDS.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
            Collections.singletonList("cd / && sudo python3 server.py
80")))
        .build();

    getSSMClient().sendCommand(sendCommandRequest);
    System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress)
{
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}
```

```
/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.instanceProfile().instanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
```

```

        DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(attachedPolicy.policyArn())
            .build();

        getIAMClient().detachRolePolicy(request);
        System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
    }

    getIAMClient().deleteRole(deleteRoleRequest);
    System.out.println("Instance profile and role deleted.");

} catch (IamException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

    getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network,
you

```

```
    * must instead specify a prefix list ID. You can also temporarily open the
port
    * to
    * any IP address while running this example. If you do, be sure to remove
    * public
    * access when you're done.
    *
    */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " +
secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " +
ipPermission);

                    for (IpRange ipRange : ipPermission.ipRanges()) {
```

```

        String cidrIp = ipRange.cidrIp();
        if (cidrIp.startsWith(ipAddress) ||
        cidrIp.equals("0.0.0.0/0")) {
            System.out.println(cidrIp + " is applicable");
            portIsOpen = true;
        }
    }

    if (!ipPermission.prefixListIds().isEmpty()) {
        System.out.println("Prefix lList is applicable");
        portIsOpen = true;
    }

    if (!portIsOpen) {
        System.out
            .println("The inbound rule does not appear to
be open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
    } else {
        break;
    }
}
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {

```

```
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
    .autoScalingGroupName(asGroupName)
    .targetGroupARNs(targetGroupARN)
    .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
    System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

    String availabilityZones = String.join(",", availabilityZoneNames);
    LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

    String[] zones = availabilityZones.split(",");
```

```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability
Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
```

```
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}
```

```
// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to
the policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
        .builder()
        .policyArn(policy.arn())
        .build();

        ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
            .listEntitiesForPolicy(listEntitiesRequest);
        if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
            || !listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
```

```
        .policyArn(policy.arn())
        .roleName(roleName) // Specify the name of the IAM
role
        .build();

        iamClient.detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
        .policyArn(policy.arn())
        .build();

    iamClient.deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(profile.getInstanceProfileName())
        .roleName(roleName) // Remove the extra dot here
        .build();

    iamClient.removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance
profile " + profile.getInstanceProfileName());
}

// Delete the instance profile after removing all roles
```

```

        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()

            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())

```

```
        .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                .build();

            getLoadBalancerClient().deleteLoadBalancer(
                builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
```

```
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load
balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
                System.out.println("Got connection error from load balancer
endpoint, retrying...");
                TimeUnit.SECONDS.sleep(15);
            }
        }
    }

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }
}
```

```
        System.out.println("Status.." + success);
        return success;
    }

    /**
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
     instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
            .healthCheckPath("/healthcheck")
            .healthCheckTimeoutSeconds(5)
            .port(port)
            .vpcId(vpcId)
            .name(targetGroupName)
            .protocol(protocol)
            .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
```

```
        .map(Subnet::subnetId)
        .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
        .subnets(subnetIdStrings)
        .name(lbName)
        .scheme("internet-facing")
        .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

        CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

        .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
```

```

        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
        + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder()

```

```

        .tableName(tableName)
        .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " +
e.getMessage());
    }
    return false;
}

/*
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended,
such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException
{
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()

```

```

        .attributeName("MediaType")
        .keyType(KeyType.HASH)
        .build(),
        KeySchemaElement.builder()
        .attributeName("ItemId")
        .keyType(KeyType.RANGE)
        .build())
    .provisionedThroughput(
        ProvisionedThroughput.builder()
        .readCapacityUnits(5L)
        .writeCapacityUnits(5L)
        .build())
    .build();

getDynamoDbClient().createTable(createTableRequest);
System.out.println("Creating table " + tableName + "...");

// Wait until the Amazon DynamoDB table is created.
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitForTableExists(tableRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Table " + tableName + " created.");

// Add records to the table.
populateTable(fileName, tableName);
}
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws
IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
}

```

```
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }
}
```

```
public void put(String name, String value) {
    SsmClient ssmClient = SsmClient.builder()
        .region(Region.US_EAST_1)
        .build();

    PutParameterRequest parameterRequest = PutParameterRequest.builder()
        .name(name)
        .value(value)
        .overwrite(true)
        .type("String")
        .build();

    ssmClient.putParameter(parameterRequest);
    System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAvailabilityZones](#)
 - [DescribeInstanceProfileAssociations](#)

- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 * - deploy
 * - demo
 * - destroy
 */
```

```

    * Each of these stages has a corresponding file prefixed with steps-*.
    */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}

```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
```

```
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
```

```
ScenarioInput,
ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
      })
    );
  })
],
```

```

        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      )),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  )),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
    */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  )),
  new ScenarioOutput(
    "populatedTable",

```

```

    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    ),
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)

```

```
        .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
    ),
    new ScenarioOutput(
        "creatingInstanceRole",
        MESSAGES.creatingInstanceRole.replace(
            "${INSTANCE_ROLE_NAME}",
            NAMES.instanceRoleName,
        ),
    ),
    new ScenarioAction("createInstanceRole", () => {
        const client = new IAMClient({});
        return client.send(
            new CreateRoleCommand({
                RoleName: NAMES.instanceRoleName,
                AssumeRolePolicyDocument: readFileSync(
                    join(ROOT, "assume-role-policy.json"),
                ),
            }),
        ),
    });
    new ScenarioOutput(
        "createdInstanceRole",
        MESSAGES.createdInstanceRole.replace(
            "${INSTANCE_ROLE_NAME}",
            NAMES.instanceRoleName,
        ),
    ),
    new ScenarioOutput(
        "attachingPolicyToRole",
        MESSAGES.attachingPolicyToRole
            .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
            .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
    ),
    new ScenarioAction("attachPolicyToRole", async (state) => {
        const client = new IAMClient({});
        await client.send(
            new AttachRolePolicyCommand({
                RoleName: NAMES.instanceRoleName,
                PolicyArn: state.instancePolicyArn,
            }),
        );
    }),
    new ScenarioOutput(
        "attachedPolicyToRole",
```

```
MESSAGES.attachedPolicyToRole
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
    ),
),
new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
        InstanceProfile: { Arn },
    } = await client.send(
        new CreateInstanceProfileCommand({
            InstanceProfileName: NAMES.instanceProfileName,
        }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
        { client },
        { InstanceProfileName: NAMES.instanceProfileName },
    );
}),
new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
),
new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
        new AddRoleToInstanceProfileCommand({
            RoleName: NAMES.instanceRoleName,
            InstanceProfileName: NAMES.instanceProfileName,
```

```

    }),
  );
}),
new ScenarioOutput(
  "addedRoleToInstanceProfile",
  MESSAGES.addedRoleToInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
...initParamsSteps,
new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
new ScenarioAction("createLaunchTemplate", async () => {
  const ssmClient = new SSMClient({});
  const { Parameter } = await ssmClient.send(
    new GetParameterCommand({
      Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
    }),
  );
  const ec2Client = new EC2Client({});
  await ec2Client.send(
    new CreateLaunchTemplateCommand({
      LaunchTemplateName: NAMES.launchTemplateName,
      LaunchTemplateData: {
        InstanceType: "t3.micro",
        ImageId: Parameter.Value,
        IamInstanceProfile: { Name: NAMES.instanceProfileName },
        UserData: readFileSync(
          join(RESOURCES_PATH, "server_startup_script.sh"),
        ).toString("base64"),
        KeyName: NAMES.keyPairName,
      },
    }),
  );
}),
new ScenarioOutput(
  "createdLaunchTemplate",
  MESSAGES.createdLaunchTemplate.replace(
    "${LAUNCH_TEMPLATE_NAME}",
    NAMES.launchTemplateName,
  ),
),
new ScenarioOutput(
  "creatingAutoScalingGroup",
  MESSAGES.creatingAutoScalingGroup.replace(

```

```

    "${AUTO_SCALING_GROUP_NAME}",
    NAMES.autoScalingGroupName,
  ),
),
new ScenarioAction("createAutoScalingGroup", async (state) => {
  const ec2Client = new EC2Client({});
  const { AvailabilityZones } = await ec2Client.send(
    new DescribeAvailabilityZonesCommand({}),
  );
  state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
  const autoScalingClient = new AutoScalingClient({});
  await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
    autoScalingClient.send(
      new CreateAutoScalingGroupCommand({
        AvailabilityZones: state.availabilityZoneNames,
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        LaunchTemplate: {
          LaunchTemplateName: NAMES.launchTemplateName,
          Version: "$Default",
        },
        MinSize: 3,
        MaxSize: 3,
      }),
    ),
  );
}),
new ScenarioOutput(
  "createdAutoScalingGroup",
  /**
   * @param {{ availabilityZoneNames: string[] }} state
   */
  (state) =>
    MESSAGES.createdAutoScalingGroup
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
      .replace(
        "${AVAILABILITY_ZONE_NAMES}",
        state.availabilityZoneNames.join(", "),
      ),
),
new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
  type: "confirm",
}),
new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),

```

```

new ScenarioAction("getVpc", async (state) => {
  const client = new EC2Client({});
  const { Vpcs } = await client.send(
    new DescribeVpcsCommand({
      Filters: [{ Name: "is-default", Values: ["true"] }],
    }),
  );
  state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
  MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
  const client = new EC2Client({});
  const { Subnets } = await client.send(
    new DescribeSubnetsCommand({
      Filters: [
        { Name: "vpc-id", Values: [state.defaultVpc] },
        { Name: "availability-zone", Values: state.availabilityZoneNames },
        { Name: "default-for-az", Values: ["true"] },
      ],
    }),
  );
  state.subnets = Subnets.map((subnet) => subnet.SubnetId);
}),
new ScenarioOutput(
  "gotSubnets",
  /**
   * @param {{ subnets: string[] }} state
   */
  (state) =>
    MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
),
new ScenarioOutput(
  "creatingLoadBalancerTargetGroup",
  MESSAGES.creatingLoadBalancerTargetGroup.replace(
    "${TARGET_GROUP_NAME}",
    NAMES.loadBalancerTargetGroupName,
  ),
),
new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(

```

```
    new CreateTargetGroupCommand({
      Name: NAMES.loadBalancerTargetGroupName,
      Protocol: "HTTP",
      Port: 80,
      HealthCheckPath: "/healthcheck",
      HealthCheckIntervalSeconds: 10,
      HealthCheckTimeoutSeconds: 5,
      HealthyThresholdCount: 2,
      UnhealthyThresholdCount: 2,
      VpcId: state.defaultVpc,
    }),
  );
  const targetGroup = TargetGroups[0];
  state.targetGroupArn = targetGroup.TargetGroupArn;
  state.targetGroupProtocol = targetGroup.Protocol;
  state.targetGroupPort = targetGroup.Port;
}),
new ScenarioOutput(
  "createdLoadBalancerTargetGroup",
  MESSAGES.createdLoadBalancerTargetGroup.replace(
    "${TARGET_GROUP_NAME}",
    NAMES.loadBalancerTargetGroupName,
  ),
),
new ScenarioOutput(
  "creatingLoadBalancer",
  MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
),
new ScenarioAction("createLoadBalancer", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
    new CreateLoadBalancerCommand({
      Name: NAMES.loadBalancerName,
      Subnets: state.subnets,
    }),
  );
  state.loadBalancerDns = LoadBalancers[0].DNSName;
  state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
  await waitUntilLoadBalancerAvailable(
    { client },
    { Names: [NAMES.loadBalancerName] },
  );
}),
new ScenarioOutput("createdLoadBalancer", (state) =>
```

```
MESSAGES.createdLoadBalancer
  .replace("${LB_NAME}", NAMES.loadBalancerName)
  .replace("${DNS_NAME}", state.loadBalancerDns),
),
new ScenarioOutput(
  "creatingListener",
  MESSAGES.creatingLoadBalancerListener
    .replace("${LB_NAME}", NAMES.loadBalancerName)
    .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
),
new ScenarioAction("createListener", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { Listeners } = await client.send(
    new CreateListenerCommand({
      LoadBalancerArn: state.loadBalancerArn,
      Protocol: state.targetGroupProtocol,
      Port: state.targetGroupPort,
      DefaultActions: [
        { Type: "forward", TargetGroupArn: state.targetGroupArn },
      ],
    })),
  );
  const listener = Listeners[0];
  state.loadBalancerListenerArn = listener.ListenerArn;
}),
new ScenarioOutput("createdListener", (state) =>
  MESSAGES.createdLoadBalancerListener.replace(
    "${LB_LISTENER_ARN}",
    state.loadBalancerListenerArn,
  ),
),
new ScenarioOutput(
  "attachingLoadBalancerTargetGroup",
  MESSAGES.attachingLoadBalancerTargetGroup
    .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
    .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
),
new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
  const client = new AutoScalingClient({});
  await client.send(
    new AttachLoadBalancerTargetGroupsCommand({
      AutoScalingGroupName: NAMES.autoScalingGroupName,
      TargetGroupARNs: [state.targetGroupArn],
    })),
  ),
```

```

    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
),

```

```
new ScenarioOutput(
  "verifiedInboundPort",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return MESSAGES.foundIpRules.replace(
        "${IP_RULES}",
        JSON.stringify(state.myIpRules, null, 2),
      );
    }
    return MESSAGES.noIpRules;
  },
),
new ScenarioInput(
  "shouldAddInboundRule",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return false;
    }
    return MESSAGES.noIpRules;
  },
  { type: "confirm" },
),
new ScenarioAction(
  "addInboundRule",
  /**
   * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup }} state
   */
  async (state) => {
    if (!state.shouldAddInboundRule) {
      return;
    }

    const client = new EC2Client({});
    await client.send(
      new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.defaultSecurityGroup.GroupId,
        CidrIp: `${state.myIp}/32`,
      })
    );
  }
)
```

```
        FromPort: 80,
        ToPort: 80,
        IpProtocol: "tcp",
    })),
    );
},
),
new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
        return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
}),
new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
),
new ScenarioAction("verifyEndpoint", async (state) => {
    try {
        const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
            axios.get(`http://${state.loadBalancerDns}`),
        );
        state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
        state.verifyEndpointError = e;
    }
}),
new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
        console.error(state.verifyEndpointError);
    } else {
        return MESSAGES.verifiedEndpoint.replace(
            "${ENDPOINT_RESPONSE}",
            state.endpointResponse,
        );
    }
}),
saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
```

```
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
```

```
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}\n`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );
});

const { TargetHealthDescriptions } = await client.send(
  new DescribeTargetHealthCommand({
    TargetGroupArn: TargetGroups[0].TargetGroupArn,
  }),
);
state.targetHealthDescriptions = TargetHealthDescriptions;
});
```

```
const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);
```

```
    },
  );

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
],
```

```
...statusSteps,
new ScenarioInput(
  "staticResponseConfirmation",
  MESSAGES.demoStaticResponseConfirmation,
  { type: "confirm" },
),
new ScenarioAction("staticResponse", async (state) => {
  if (!state.staticResponseConfirmation) {
    process.exit();
  } else {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmFailureResponseKey,
        Value: "static",
        Overwrite: true,
        Type: "String",
      }),
    );
  }
}),
new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
...statusSteps,
new ScenarioInput(
  "badCredentialsConfirmation",
  MESSAGES.demoBadCredentialsConfirmation,
  { type: "confirm" },
),
new ScenarioAction("badCredentialsExit", (state) => {
  if (!state.badCredentialsConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("fixDynamoDBName", async () => {
  const client = new SSMClient({});
  await client.send(
    new PutParameterCommand({
      Name: NAMES.ssmTableNameKey,
      Value: NAMES.tableName,
      Overwrite: true,
      Type: "String",
    }),
  );
}),
}),
```

```
new ScenarioAction(
  "badCredentials",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
   */
  async (state) => {
    await createSsmOnlyInstanceProfile();
    const autoScalingClient = new AutoScalingClient({});
    const { AutoScalingGroups } = await autoScalingClient.send(
      new DescribeAutoScalingGroupsCommand({
        AutoScalingGroupNames: [NAMES.autoScalingGroupName],
      }),
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );
    });
  }
);
```

```
    );

    const instance = InstanceInformationList.find(
      (info) => info.InstanceId === state.targetInstance.InstanceId,
    );

    if (!instance) {
      throw new Error("Instance not found.");
    }
  });

  await ssmClient.send(
    new SendCommandCommand({
      InstanceIds: [state.targetInstance.InstanceId],
      DocumentName: "AWS-RunShellScript",
      Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
    }),
  );
},
),
new ScenarioOutput(
  "testBadCredentials",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
   ssm').InstanceInformation}} state
   */
  (state) =>
    MESSAGES.demoTestBadCredentials.replace(
      "${INSTANCE_ID}",
      state.targetInstance.InstanceId,
    ),
),
loadBalancerLoop,
new ScenarioInput(
  "deepHealthCheckConfirmation",
  MESSAGES.demoDeepHealthCheckConfirmation,
  { type: "confirm" },
),
new ScenarioAction("deepHealthCheckExit", (state) => {
  if (!state.deepHealthCheckConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("deepHealthCheck", async () => {
```

```

const client = new SSMClient({});
await client.send(
  new PutParameterCommand({
    Name: NAMES.ssmHealthCheckKey,
    Value: "deep",
    Overwrite: true,
    Type: "String",
  }),
);
}),
new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput(
  "killInstanceConfirmation",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
   */
  (state) =>
    MESSAGES.demoKillInstanceConfirmation.replace(
      "${INSTANCE_ID}",
      state.targetInstance.InstanceId,
    ),
  { type: "confirm" },
),
new ScenarioAction("killInstanceExit", (state) => {
  if (!state.killInstanceConfirmation) {
    process.exit();
  }
}),
new ScenarioAction(
  "killInstance",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
   */
  async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new TerminateInstanceInAutoScalingGroupCommand({
        InstanceId: state.targetInstance.InstanceId,
        ShouldDecrementDesiredCapacity: false,
      }),
    ),

```

```
    );
  },
),
new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
  type: "confirm",
}),
new ScenarioAction("failOpenExit", (state) => {
  if (!state.failOpenConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("failOpen", () => {
  const client = new SSMClient({});
  return client.send(
    new PutParameterCommand({
      Name: NAMES.ssmTableNameKey,
      Value: `fake-table-${Date.now()}`,
      Overwrite: true,
      Type: "String",
    }),
  ),
);
}),
new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput(
  "resetTableConfirmation",
  MESSAGES.demoResetTableConfirmation,
  { type: "confirm" },
),
new ScenarioAction("resetTableExit", (state) => {
  if (!state.resetTableConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("resetTable", async () => {
  const client = new SSMClient({});
  await client.send(
    new PutParameterCommand({
      Name: NAMES.ssmTableNameKey,
      Value: NAMES.tableName,
```

```
        Overwrite: true,
        Type: "String",
    })),
    );
}),
new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
healthCheckLoop,
loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
    const iamClient = new IAMClient({});
    const { Policy } = await iamClient.send(
        new CreatePolicyCommand({
            PolicyName: NAMES.ssmOnlyPolicyName,
            PolicyDocument: readFileSync(
                join(RESOURCES_PATH, "ssm_only_policy.json"),
            ),
        })),
    );
    await iamClient.send(
        new CreateRoleCommand({
            RoleName: NAMES.ssmOnlyRoleName,
            AssumeRolePolicyDocument: JSON.stringify({
                Version: "2012-10-17",
                Statement: [
                    {
                        Effect: "Allow",
                        Principal: { Service: "ec2.amazonaws.com" },
                        Action: "sts:AssumeRole",
                    },
                ],
            })),
    );
    await iamClient.send(
        new AttachRolePolicyCommand({
            RoleName: NAMES.ssmOnlyRoleName,
            PolicyArn: Policy.Arn,
        })),
    );
    await iamClient.send(
        new AttachRolePolicyCommand({
            RoleName: NAMES.ssmOnlyRoleName,
```

```

        PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    )),
    );
    const { InstanceProfile } = await iamClient.send(
        new CreateInstanceProfileCommand({
            InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        })),
    );
    await waitUntilInstanceProfileExists(
        { client: iamClient },
        { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
    );
    await iamClient.send(
        new AddRoleToInstanceProfileCommand({
            InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
            RoleName: NAMES.ssmOnlyRoleName,
        })),
    );

    return InstanceProfile;
}

```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```

import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
    EC2Client,
    DeleteKeyPairCommand,
    DeleteLaunchTemplateCommand,
    RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
    IAMClient,
    DeleteInstanceProfileCommand,
    RemoveRoleFromInstanceProfileCommand,
    DeletePolicyCommand,
    DeleteRoleCommand,
    DetachRolePolicyCommand,
    paginateListPolicies,
} from "@aws-sdk/client-iam";

```

```
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
```

```
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  })),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  })),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  })),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`
        );
      } else {
        await client.send(
```

```
        new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
        }),
    );
}
} catch (e) {
    state.detachPolicyFromRoleError = e;
}
}),
new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
        console.error(state.detachPolicyFromRoleError);
        return MESSAGES.detachPolicyFromRoleError
            .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
            .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
}),
new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
        state.deletePolicyError = new Error(
            `Policy ${NAMES.instancePolicyName} not found.`
        );
    } else {
        return client.send(
            new DeletePolicyCommand({
                PolicyArn: policy.Arn,
            }),
        );
    }
}),
new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
        console.error(state.deletePolicyError);
        return MESSAGES.deletePolicyError.replace(
            "${INSTANCE_POLICY_NAME}",
            NAMES.instancePolicyName,
        );
    }
});
```

```
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }
},
new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new RemoveRoleFromInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  } catch (e) {
    state.removeRoleFromInstanceProfileError = e;
  }
}),
new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
  if (state.removeRoleFromInstanceProfile) {
    console.error(state.removeRoleFromInstanceProfileError);
    return MESSAGES.removeRoleFromInstanceProfileError
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }
  return MESSAGES.removedRoleFromInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
}),
new ScenarioAction("deleteInstanceRole", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new DeleteRoleCommand({
        RoleName: NAMES.instanceRoleName,
      }),
    );
  } catch (e) {
    state.deleteInstanceRoleError = e;
  }
}),
new ScenarioOutput("deleteInstanceRoleResult", (state) => {
  if (state.deleteInstanceRoleError) {
```

```
    console.error(state.deleteInstanceRoleError);
    return MESSAGES.deleteInstanceRoleError.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }
  return MESSAGES.deletedInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  );
}),
new ScenarioAction("deleteInstanceProfile", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new DeleteInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  } catch (e) {
    state.deleteInstanceProfileError = e;
  }
}),
new ScenarioOutput("deleteInstanceProfileResult", (state) => {
  if (state.deleteInstanceProfileError) {
    console.error(state.deleteInstanceProfileError);
    return MESSAGES.deleteInstanceProfileError.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }
  return MESSAGES.deletedInstanceProfile.replace(
    "${INSTANCE_PROFILE_NAME}",
    NAMES.instanceProfileName,
  );
}),
new ScenarioAction("deleteAutoScalingGroup", async (state) => {
  try {
    await terminateGroupInstances(NAMES.autoScalingGroupName);
    await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
      await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
    });
  } catch (e) {
    state.deleteAutoScalingGroupError = e;
  }
});
```

```
    }
  })),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  })),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  })),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  })),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
```

```
    await client.send(
      new DeleteLoadBalancerCommand({
        LoadBalancerArn: loadBalancer.LoadBalancerArn,
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
      const lb = await findLoadBalancer(NAMES.loadBalancerName);
      if (lb) {
        throw new Error("Load balancer still exists.");
      }
    });
  } catch (e) {
    state.deleteLoadBalancerError = e;
  }
}),
new ScenarioOutput("deleteLoadBalancerResult", (state) => {
  if (state.deleteLoadBalancerError) {
    console.error(state.deleteLoadBalancerError);
    return MESSAGES.deleteLoadBalancerError.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }
  return MESSAGES.deletedLoadBalancer.replace(
    "${LB_NAME}",
    NAMES.loadBalancerName,
  );
}),
new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  try {
    const { TargetGroups } = await client.send(
      new DescribeTargetGroupsCommand({
        Names: [NAMES.loadBalancerTargetGroupName],
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      client.send(
        new DeleteTargetGroupCommand({
          TargetGroupArn: TargetGroups[0].TargetGroupArn,
        }),
      ),
    );
  }
});
```

```
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  })),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  })),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        })),
    );
  } catch (e) {
    state.detachSsmOnlyRoleFromProfileError = e;
  }
  })),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  })),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
```

```
    const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
    await iamClient.send(
      new DetachRolePolicyCommand({
        RoleName: NAMES.ssmOnlyRoleName,
        PolicyArn: ssmOnlyPolicy.Arn,
      }),
    );
  } catch (e) {
    state.detachSsmOnlyCustomRolePolicyError = e;
  }
}),
new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
  if (state.detachSsmOnlyCustomRolePolicyError) {
    console.error(state.detachSsmOnlyCustomRolePolicyError);
    return MESSAGES.detachSsmOnlyCustomRolePolicyError
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }
  return MESSAGES.detachedSsmOnlyCustomRolePolicy
    .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
    .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
}),
new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
  try {
    const iamClient = new IAMClient({});
    await iamClient.send(
      new DetachRolePolicyCommand({
        RoleName: NAMES.ssmOnlyRoleName,
        PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
      }),
    );
  } catch (e) {
    state.detachSsmOnlyAWSRolePolicyError = e;
  }
}),
new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
  if (state.detachSsmOnlyAWSRolePolicyError) {
    console.error(state.detachSsmOnlyAWSRolePolicyError);
    return MESSAGES.detachSsmOnlyAWSRolePolicyError
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }
  return MESSAGES.detachedSsmOnlyAWSRolePolicy
    .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
```

```
    .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }},
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        })),
    );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }},
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }},
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        })),
    );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }},
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
```

```

        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
    );
}
return MESSAGES.deletedSsmOnlyPolicy.replace(
    "${POLICY_NAME}",
    NAMES.ssmOnlyPolicyName,
);
}),
new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
        const iamClient = new IAMClient({});
        await iamClient.send(
            new DeleteRoleCommand({
                RoleName: NAMES.ssmOnlyRoleName,
            }),
        );
    } catch (e) {
        state.deleteSsmOnlyRoleError = e;
    }
}),
new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
        console.error(state.deleteSsmOnlyRoleError);
        return MESSAGES.deleteSsmOnlyRoleError.replace(
            "${ROLE_NAME}",
            NAMES.ssmOnlyRoleName,
        );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
    );
}),
new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
        /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
        state,
    ) => {
        const ec2Client = new EC2Client({});

        try {
            await ec2Client.send(

```

```

        new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
        }),
    );
} catch (e) {
    state.revokeSecurityGroupIngressError = e;
}
},
),
new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
        console.error(state.revokeSecurityGroupIngressError);
        return MESSAGES.revokeSecurityGroupIngressError.replace(
            "${IP}",
            state.myIp,
        );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
}),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
    const client = new IAMClient({});
    const paginatedPolicies = paginateListPolicies({ client }, {});
    for await (const page of paginatedPolicies) {
        const policy = page.Policies.find((p) => p.PolicyName === policyName);
        if (policy) {
            return policy;
        }
    }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
    const client = new AutoScalingClient({});

```

```
try {
  await client.send(
    new DeleteAutoScalingGroupCommand({
      AutoScalingGroupName: groupName,
    }),
  );
} catch (err) {
  if (!(err instanceof Error)) {
    throw err;
  }
  console.log(err.name);
  throw err;
}
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
```

```
    (g) => g.AutoScalingGroupName === groupName,
  );
  if (group) {
    return group;
  }
}
throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAvailabilityZones](#)
 - [DescribeIamInstanceProfileAssociations](#)
 - [DescribeInstances](#)
 - [DescribeLoadBalancers](#)
 - [DescribeSubnets](#)
 - [DescribeTargetGroups](#)
 - [DescribeTargetHealth](#)
 - [DescribeVpcs](#)
 - [RebootInstances](#)

- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
    resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
        such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
```

```
self.recommendation = recommendation
self.autoscaler = autoscaler
self.loadbalancer = loadbalancer
self.param_helper = param_helper
self.protocol = "HTTP"
self.port = 80
self.ssh_port = 22

prefix = "doc-example-resilience"
self.target_group_name = f"{prefix}-tg"
self.load_balancer_name = f"{prefix}-lb"

def deploy(self) -> None:
    """
    Deploys the resources required for the resilient service, including the
    DynamoDB table,
    EC2 instances, Auto Scaling group, and load balancer.
    """
    recommendations_path = f"{self.resource_path}/recommendations.json"
    startup_script = f"{self.resource_path}/server_startup_script.sh"
    instance_policy = f"{self.resource_path}/instance_policy.json"

    logging.info("Starting deployment of resources for the resilient
service.")

    logging.info(
        "Creating and populating DynamoDB table '%s'.",
        self.recommendation.table_name,
    )
    self.recommendation.create()
    self.recommendation.populate(recommendations_path)

    logging.info(
        "Creating an EC2 launch template with the startup script '%s'.",
        startup_script,
    )
    self.autoscaler.create_template(startup_script, instance_policy)

    logging.info(
        "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
    )
    zones = self.autoscaler.create_autoscaling_group(3)
```

```
logging.info("Creating variables that control the flow of the demo.")
self.param_helper.reset()

logging.info("Creating Elastic Load Balancing target group and load
balancer.")

vpc = self.autoscaler.get_default_vpc()
subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
target_group = self.loadbalancer.create_target_group(
    self.target_group_name, self.protocol, self.port, vpc["VpcId"]
)
self.loadbalancer.create_load_balancer(
    self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
)
self.loadbalancer.create_listener(self.load_balancer_name, target_group)

self.autoscaler.attach_load_balancer_target_group(target_group)

logging.info("Verifying access to the load balancer endpoint.")
endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

if not lb_success:
    logging.warning(
        "Couldn't connect to the load balancer. Verifying that the port
is open..."
    )
    sec_group, port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.port, current_ip_address
    )
    sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.ssh_port, current_ip_address
    )
    if not port_is_open:
        logging.warning(
            "The default security group for your VPC must allow access
from this computer."
        )
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
```

```

        f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
        q.is_yesno,
    ):
        self.autoscaler.open_inbound_port(
            sec_group["GroupId"], self.port, current_ip_address
        )
    if not ssh_port_is_open:
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
            f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
            q.is_yesno,
        ):
            self.autoscaler.open_inbound_port(
                sec_group["GroupId"], self.ssh_port, current_ip_address
            )
        lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

    if lb_success:
        logging.info(
            "Load balancer is ready. Access it at: http://%s",
current_ip_address
        )
    else:
        logging.error(
            "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
        )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0

```

```
    while choice != 2:
        logging.info("Choose an action to interact with the service.")
        choice = q.choose("Which action would you like to take? ", actions)
        if choice == 0:
            logging.info("Sending a GET request to the load balancer
endpoint.")
            endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
            logging.info("GET http://%s", endpoint)
            response = requests.get(f"http://{endpoint}")
            logging.info("Response: %s", response.status_code)
            if response.headers.get("content-type") == "application/json":
                pp(response.json())
        elif choice == 1:
            logging.info("Checking the health of load balancer targets.")
            health =
self.loadbalancer.check_target_health(self.target_group_name)
            for target in health:
                state = target["TargetHealth"]["State"]
                logging.info(
                    "Target %s on port %d is %s",
                    target["Target"]["Id"],
                    target["Target"]["Port"],
                    state,
                )
                if state != "healthy":
                    logging.warning(
                        "%s: %s",
                        target["TargetHealth"]["Reason"],
                        target["TargetHealth"]["Description"],
                    )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
        elif choice == 2:
            logging.info("Proceeding to the next part of the demo.")

def demo(self) -> None:
    """
    Runs the demonstration, showing how the service responds to different
failure scenarios
    and how a resilient architecture can keep the service running.
    """
```

```
ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

logging.info("Resetting parameters to starting values for the demo.")
self.param_helper.reset()

logging.info(
    "Starting demonstration of the service's resilience under various
failure conditions."
)
self.demo_choices()

logging.info(
    "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
)
self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
logging.info("Sending GET requests will now return failure codes.")
self.demo_choices()

logging.info("Switching to static response mode to mitigate failure.")
self.param_helper.put(self.param_helper.failure_response, "static")
logging.info("Sending GET requests will now return static responses.")
self.demo_choices()

logging.info("Restoring normal operation of the recommendation service.")
self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

logging.info(
    "Introducing a failure by assigning bad credentials to one of the
instances."
)
self.autoscaler.create_instance_profile(
    ssm_only_policy,
    self.autoscaler.bad_creds_policy_name,
    self.autoscaler.bad_creds_role_name,
    self.autoscaler.bad_creds_profile_name,
    ["AmazonSSMManagedInstanceCore"],
)
instances = self.autoscaler.get_instances()
bad_instance_id = instances[0]
instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
logging.info(
    "Replacing instance profile with bad credentials for instance %s.",
```

```
        bad_instance_id,
    )
    self.autoscaler.replace_instance_profile(
        bad_instance_id,
        self.autoscaler.bad_creds_profile_name,
        instance_profile["AssociationId"],
    )
    logging.info(
        "Sending GET requests may return either a valid recommendation or a
static response."
    )
    self.demo_choices()

    logging.info("Implementing deep health checks to detect unhealthy
instances.")
    self.param_helper.put(self.param_helper.health_check, "deep")
    logging.info("Checking the health of the load balancer targets.")
    self.demo_choices()

    logging.info(
        "Terminating the unhealthy instance to let the auto scaler replace
it."
    )
    self.autoscaler.terminate_instance(bad_instance_id)
    logging.info("The service remains resilient during instance
replacement.")
    self.demo_choices()

    logging.info("Simulating a complete failure of the recommendation
service.")
    self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
    logging.info(
        "All instances will report as unhealthy, but the service will still
return static responses."
    )
    self.demo_choices()
    self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
```

```
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
    if automation:
        cleanup = True
    else:
        cleanup = q.ask(
            "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
        )

    if cleanup:
        logging.info("Deleting load balancer and related resources.")
        self.loadbalancer.delete_load_balancer(self.load_balancer_name)
        self.loadbalancer.delete_target_group(self.target_group_name)
        self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
        self.autoscaler.delete_key_pair()
        self.autoscaler.delete_template()
        self.autoscaler.delete_instance_profile(
            self.autoscaler.bad_creds_profile_name,
            self.autoscaler.bad_creds_role_name,
        )
        logging.info("Deleting DynamoDB table and other resources.")
        self.recommendation.destroy()
    else:
        logging.warning(
            "Resources have not been deleted. Ensure you clean them up
manually to avoid unexpected charges."
        )

def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
```

```
)
parser.add_argument(
    "--resource_path",
    default="../../../../scenarios/features/resilient_service/resources",
    help="The path to resource files used by this example, such as IAM
policies and\n"
    "instance scripts.",
)
args = parser.parse_args()

logging.info("Starting the Resilient Service demo.")

prefix = "doc-example-resilience"

# Service Clients
ddb_client = boto3.client("dynamodb")
elb_client = boto3.client("elbv2")
autoscaling_client = boto3.client("autoscaling")
ec2_client = boto3.client("ec2")
ssm_client = boto3.client("ssm")
iam_client = boto3.client("iam")

# Wrapper instantiations
recommendation = RecommendationService(
    "doc-example-recommendation-service", ddb_client
)
autoscaling_wrapper = AutoScalingWrapper(
    prefix,
    "t3.micro",
    "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
    autoscaling_client,
    ec2_client,
    ssm_client,
    iam_client,
)
elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
param_helper = ParameterHelper(recommendation.table_name, ssm_client)

# Demo invocation
runner = Runner(
    args.resource_path,
    recommendation,
    autoscaling_wrapper,
    elb_wrapper,
```

```
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
```

```

        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
        definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:

```

```

        response = self.iam_client.create_policy(
            PolicyName=policy_name, PolicyDocument=policy_doc
        )
        policy_arn = response["Policy"]["Arn"]
        log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
        return policy_arn

    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            # If the policy already exists, get its ARN
            response = self.iam_client.get_policy(
                PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
            return policy_arn
        log.error(f"Full error:\n\t{err}")

def create_role(self, role_name: str, assume_role_doc: dict) -> str:
    """
    Creates a new IAM role or retrieves the ARN of an existing role.

    :param role_name: The name to give the created role.
    :param assume_role_doc: The assume role policy document that specifies
which
                            entities can assume the role.
    :return: The ARN of the created or existing role.
    """
    try:
        response = self.iam_client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(assume_role_doc)
        )
        role_arn = response["Role"]["Arn"]
        log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
        return role_arn

    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            # If the role already exists, get its ARN
            response = self.iam_client.get_role(RoleName=role_name)

```

```

        role_arn = response["Role"]["Arn"]
        log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
        return role_arn
    log.error(f"Full error:\n\t{err}")

def attach_policy(
    self,
    role_name: str,
    policy_arn: str,
    aws_managed_policies: Tuple[str, ...] = (),
) -> None:
    """
    Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

    :param role_name: The name of the role to attach the policy to.
    :param policy_arn: The ARN of the policy to attach.
    :param aws_managed_policies: A tuple of AWS-managed policy names to
attach to the role.
    """
    try:
        self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
        for aws_policy in aws_managed_policies:
            self.iam_client.attach_role_policy(
                RoleName=role_name,
                PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
            )
        log.info(f"Attached policy {policy_arn} to role {role_name}.")
    except ClientError as err:
        log.error(f"Failed to attach policy {policy_arn} to role
{role_name}.")
        log.error(f"Full error:\n\t{err}")

def create_instance_profile(
    self,
    policy_file: str,
    policy_name: str,
    role_name: str,
    profile_name: str,
    aws_managed_policies: Tuple[str, ...] = (),
) -> str:
    """

```

Creates a policy, role, and profile that is associated with instances created by this class. An instance's associated profile defines a role that is assumed by the instance. The role has attached policies that specify the AWS permissions granted to clients that run on the instance.

:param policy_file: The name of a JSON file that contains the policy definition to

create and attach to the role.

:param policy_name: The name to give the created policy.

:param role_name: The name to give the created role.

:param profile_name: The name to the created profile.

:param aws_managed_policies: Additional AWS-managed policies that are attached to

the role, such as

AmazonSSMManagedInstanceCore to grant

use of Systems Manager to send commands to

the instance.

:return: The ARN of the profile that is created.

"""

```
assume_role_doc = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Service": "ec2.amazonaws.com"},
            "Action": "sts:AssumeRole",
        }
    ],
}

policy_arn = self.create_policy(policy_file, policy_name)
self.create_role(role_name, assume_role_doc)
self.attach_policy(role_name, policy_arn, aws_managed_policies)

try:
    profile_response = self.iam_client.create_instance_profile(
        InstanceProfileName=profile_name
    )
    waiter = self.iam_client.get_waiter("instance_profile_exists")
    waiter.wait(InstanceProfileName=profile_name)
    time.sleep(10) # wait a little longer
    profile_arn = profile_response["InstanceProfile"]["Arn"]
```

```

        self.iam_client.add_role_to_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )
        log.info("Created profile %s and added role %s.", profile_name,
role_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            prof_response = self.iam_client.get_instance_profile(
                InstanceProfileName=profile_name
            )
            profile_arn = prof_response["InstanceProfile"]["Arn"]
            log.info(
                "Instance profile %s already exists, nothing to do.",
profile_name
            )
            log.error(f"Full error:\n\t{err}")
        return profile_arn

def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
        if not response["IamInstanceProfileAssociations"]:
            log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
    except ClientError as err:
        log.error(
            f"Failed to retrieve instance profile for instance
{instance_id}."
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidInstanceID.NotFound":

```

```
        log.error(f"The instance ID '{instance_id}' does not exist.")
        log.error(f"Full error:\n\t{err}")

def replace_instance_profile(
    self,
    instance_id: str,
    new_instance_profile_name: str,
    profile_association_id: str,
) -> None:
    """
    Replaces the profile associated with a running instance. After the
    profile is
    replaced, the instance is rebooted to ensure that it uses the new
    profile. When
    the instance is ready, Systems Manager is used to restart the Python web
    server.

    :param instance_id: The ID of the instance to restart.
    :param new_instance_profile_name: The name of the new profile to
    associate with
                               the specified instance.
    :param profile_association_id: The ID of the existing profile association
    for the
                               instance.
    """
    try:
        self.ec2_client.replace_iam_instance_profile_association(
            IamInstanceProfile={"Name": new_instance_profile_name},
            AssociationId=profile_association_id,
        )
        log.info(
            "Replaced instance profile for association %s with profile %s.",
            profile_association_id,
            new_instance_profile_name,
        )
        time.sleep(5)

        self.ec2_client.reboot_instances(InstanceIds=[instance_id])
        log.info("Rebooting instance %s.", instance_id)
        waiter = self.ec2_client.get_waiter("instance_running")
        log.info("Waiting for instance %s to be running.", instance_id)
        waiter.wait(InstanceIds=[instance_id])
        log.info("Instance %s is now running.", instance_id)
```

```

        self.ssm_client.send_command(
            InstanceIds=[instance_id],
            DocumentName="AWS-RunShellScript",
            Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
        )
        log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
    except ClientError as err:
        log.error("Failed to replace instance profile.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidAssociationID.NotFound":
            log.error(
                f"Association ID '{profile_association_id}' does not exist."
                "Please check the association ID and try again."
            )
        if error_code == "InvalidInstanceId":
            log.error(
                f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                f"Please verify the instance ID and try again."
            )
        log.error(f"Full error:\n\t{err}")

def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
    """
    Detaches a role from an instance profile, detaches policies from the
role,
and deletes all the resources.

:param profile_name: The name of the profile to delete.
:param role_name: The name of the role to delete.
    """
    try:
        self.iam_client.remove_role_from_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
        log.info("Deleted instance profile %s.", profile_name)
        attached_policies = self.iam_client.list_attached_role_policies(
            RoleName=role_name
        )

```

```

        for pol in attached_policies["AttachedPolicies"]:
            self.iam_client.detach_role_policy(
                RoleName=role_name, PolicyArn=pol["PolicyArn"]
            )
            if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
            log.info("Detached and deleted policy %s.", pol["PolicyName"])
        self.iam_client.delete_role(RoleName=role_name)
        log.info("Deleted role %s.", role_name)
    except ClientError as err:
        log.error(
            f"Couldn't delete instance profile {profile_name} or detach "
            f"policies and delete role {role_name}: {err}"
        )
        if err.response["Error"]["Code"] == "NoSuchEntity":
            log.info(
                "Instance profile %s doesn't exist, nothing to do.",
profile_name
            )

def create_key_pair(self, key_pair_name: str) -> None:
    """
    Creates a new key pair.

    :param key_pair_name: The name of the key pair to create.
    """
    try:
        response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
        with open(f"{key_pair_name}.pem", "w") as file:
            file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
        log.info("Created key pair %s.", key_pair_name)
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to create key pair {key_pair_name}.")
        if error_code == "InvalidKeyPair.Duplicate":
            log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
        log.error(f"Full error:\n\t{err}")

def delete_key_pair(self) -> None:
    """

```

```

Deletes a key pair.
"""
try:
    self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
    remove(f"{self.key_pair_name}.pem")
    log.info("Deleted key pair %s.", self.key_pair_name)
except ClientError as err:
    log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
    log.error(f"Full error:\n\t{err}")
except FileNotFoundError as err:
    log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
    log.error(f"Full error:\n\t{err}")

def create_template(
    self, server_startup_script_file: str, instance_policy_file: str
) -> Dict[str, Any]:
    """
    Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
launch template specifies a Bash script in its user data field that runs
after
the instance is started. This script installs Python packages and starts
a
Python web server on the instance.

:param server_startup_script_file: The path to a Bash script file that is
run
when an instance starts.
:param instance_policy_file: The path to a file that defines a
permissions policy
to create and attach to the instance
profile.
:return: Information about the newly created template.
"""
    template = {}
    try:
        # Create key pair and instance profile
        self.create_key_pair(self.key_pair_name)
        self.create_instance_profile(
            instance_policy_file,
            self.instance_policy_name,
            self.instance_role_name,

```

```
        self.instance_profile_name,
    )

    # Read the startup script
    with open(server_startup_script_file) as file:
        start_server_script = file.read()

    # Get the latest AMI ID
    ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
    ami_id = ami_latest["Parameter"]["Value"]

    # Create the launch template
    lt_response = self.ec2_client.create_launch_template(
        LaunchTemplateName=self.launch_template_name,
        LaunchTemplateData={
            "InstanceType": self.inst_type,
            "ImageId": ami_id,
            "IamInstanceProfile": {"Name": self.instance_profile_name},
            "UserData": base64.b64encode(
                start_server_script.encode(encoding="utf-8")
            ).decode(encoding="utf-8"),
            "KeyName": self.key_pair_name,
        },
    )
    template = lt_response["LaunchTemplate"]
    log.info(
        f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
    )
    except ClientError as err:
        log.error(f"Failed to create launch template
{self.launch_template_name}.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
            log.info(
                f"Launch template {self.launch_template_name} already exists,
nothing to do."
            )
        log.error(f"Full error:\n\t{err}")
    return template

def delete_template(self):
    """
```

```
Deletes a launch template.
"""
try:
    self.ec2_client.delete_launch_template(
        LaunchTemplateName=self.launch_template_name
    )
    self.delete_instance_profile(
        self.instance_profile_name, self.instance_role_name
    )
    log.info("Launch template %s deleted.", self.launch_template_name)
except ClientError as err:
    if (
        err.response["Error"]["Code"]
        == "InvalidLaunchTemplateName.NotFoundException"
    ):
        log.info(
            "Launch template %s does not exist, nothing to do.",
            self.launch_template_name,
        )
    log.error(f"Full error:\n\t{err}")

def get_availability_zones(self) -> List[str]:
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
    except ClientError as err:
        log.error("Failed to retrieve availability zones.")
        log.error(f"Full error:\n\t{err}")
    else:
        return zones

def create_autoscaling_group(self, group_size: int) -> List[str]:
    """
    Creates an EC2 Auto Scaling group with the specified size.
```

```
        :param group_size: The number of instances to set for the minimum and
maximum in
                the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones

    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
```

```

        AutoScalingGroupNames=[self.group_name]
    )
    instance_ids = [
        i["InstanceId"]
        for i in as_response["AutoScalingGroups"][0]["Instances"]
    ]
    log.info(
        f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
    )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
        )
        if error_code == "ResourceNotFound":
            log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
        log.error(f"Full error:\n\t{err}")
    else:
        return instance_ids

def terminate_instance(self, instance_id: str, decrementssetting=False) ->
None:
    """
    Terminates an instance in an EC2 Auto Scaling group. After an instance is
    terminated, it can no longer be accessed.

    :param instance_id: The ID of the instance to terminate.
    :param decrementssetting: If True, do not replace terminated instances.
    """
    try:
        self.autoscaling_client.terminate_instance_in_auto_scaling_group(
            InstanceId=instance_id,
            ShouldDecrementDesiredCapacity=decrementssetting,
        )
        log.info("Terminated instance %s.", instance_id)

        # Adding a waiter to ensure the instance is terminated
        waiter = self.ec2_client.get_waiter("instance_terminated")
        log.info("Waiting for instance %s to be terminated...", instance_id)
        waiter.wait(InstanceIds=[instance_id])

```

```

        log.info(
            f"Instance '{instance_id}' has been terminated and will be
replaced."
        )

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to terminate instance '{instance_id}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to terminate the instance again."
            )
        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the resource."
            )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )

```

```

    )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to attach load balancer target group
            '{lb_target_group['TargetGroupName']}'."
        )
        if error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
                the resource."
            )
        elif error_code == "ServiceLinkedRoleFailure":
            log.error(
                "The operation failed because the service-linked role is not
                ready or does not exist. "
                "Check that the service-linked role exists and is correctly
                configured."
            )
        log.error(f"Full error:\n\t{err}")

def delete_autoscaling_group(self, group_name: str) -> None:
    """
    Terminates all instances in the group, then deletes the EC2 Auto Scaling
    group.

    :param group_name: The name of the group to delete.
    """
    try:
        response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[group_name]
        )
        groups = response.get("AutoScalingGroups", [])
        if len(groups) > 0:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, MinSize=0
            )
            instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
            for inst_id in instance_ids:
                self.terminate_instance(inst_id)

```

```
        # Wait for all instances to be terminated
        if instance_ids:
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for all instances to be terminated...")
            waiter.wait(InstanceIds=instance_ids)
            log.info("All instances have been terminated.")
        else:
            log.info(f"No groups found named '{group_name}'! Nothing to do.")
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to delete the group again."
            )
        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the group."
            )
        log.error(f"Full error:\n\t{err}")

def get_default_vpc(self) -> Dict[str, Any]:
    """
    Gets the default VPC for the account.

    :return: Data about the default VPC.
    """
    try:
        response = self.ec2_client.describe_vpcs(
            Filters=[{"Name": "is-default", "Values": ["true"]}])
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error("Failed to retrieve the default VPC.")
        if error_code == "UnauthorizedOperation":
            log.error(
                "You do not have the necessary permissions to describe VPCs."
            )
    "
```

```

        "Ensure that your AWS IAM user or role has the correct
permissions."
    )
    elif error_code == "InvalidParameterValue":
        log.error(
            "One or more parameters are invalid. Check the request
parameters."
        )

        log.error(f"Full error:\n\t{err}")
    else:
        if "Vpcs" in response and response["Vpcs"]:
            log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
            return response["Vpcs"][0]
        else:
            pass

def verify_inbound_port(
    self, vpc: Dict[str, Any], port: int, ip_address: str
) -> Tuple[Dict[str, Any], bool]:
    """
    Verify the default security group of the specified VPC allows ingress
from this
    computer. This can be done by allowing ingress from this computer's IP
address. In some situations, such as connecting from a corporate network,
you
    must instead specify a prefix list ID. You can also temporarily open the
port to
    any IP address while running this example. If you do, be sure to remove
public
    access when you're done.

    :param vpc: The VPC used by this example.
    :param port: The port to verify.
    :param ip_address: This computer's IP address.
    :return: The default security group of the specified VPC, and a value
that indicates
           whether the specified port is open.
    """
    try:
        response = self.ec2_client.describe_security_groups(
            Filters=[

```

```

        {"Name": "group-name", "Values": ["default"]},
        {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
    ]
)
sec_group = response["SecurityGroups"][0]
port_is_open = False
log.info(f"Found default security group {sec_group['GroupId']}.")

for ip_perm in sec_group["IpPermissions"]:
    if ip_perm.get("FromPort", 0) == port:
        log.info(f"Found inbound rule: {ip_perm}")
        for ip_range in ip_perm["IpRanges"]:
            cidr = ip_range.get("CidrIp", "")
            if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                port_is_open = True
        if ip_perm["PrefixListIds"]:
            port_is_open = True
        if not port_is_open:
            log.info(
                f"The inbound rule does not appear to be open to
either this computer's IP "
                f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
            )
        else:
            break
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
    )
    if error_code == "InvalidVpcID.NotFound":
        log.error(
            f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
        )
        log.error(f"Full error:\n\t{err}")
    else:
        return sec_group, port_is_open

def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:

```

```
"""
Add an ingress rule to the specified security group that allows access on
the
specified port from the specified IP address.

:param sec_group_id: The ID of the security group to modify.
:param port: The port to open.
:param ip_address: The IP address that is granted access.
"""
try:
    self.ec2_client.authorize_security_group_ingress(
        GroupId=sec_group_id,
        CidrIp=f"{ip_address}/32",
        FromPort=port,
        ToPort=port,
        IpProtocol="tcp",
    )
    log.info(
        "Authorized ingress to %s on port %s from %s.",
        sec_group_id,
        port,
        ip_address,
    )
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
    )
    if error_code == "InvalidGroupId.Malformed":
        log.error(
            "The security group ID is malformed. "
            "Please verify that the security group ID is correct."
        )
    elif error_code == "InvalidPermission.Duplicate":
        log.error(
            "The specified rule already exists in the security group. "
            "Check the existing rules for this security group."
        )
    log.error(f"Full error:\n\t{err}")

def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
```

```
"""
Gets the default subnets in a VPC for a specified list of Availability
Zones.

:param vpc_id: The ID of the VPC to look up.
:param zones: The list of Availability Zones to look up.
:return: The list of subnets found.
"""
# Ensure that 'zones' is a list, even if None is passed
if zones is None:
    zones = []
try:
    paginator = self.ec2_client.get_paginator("describe_subnets")
    page_iterator = paginator.paginate(
        Filters=[
            {"Name": "vpc-id", "Values": [vpc_id]},
            {"Name": "availability-zone", "Values": zones},
            {"Name": "default-for-az", "Values": ["true"]},
        ]
    )

    subnets = []
    for page in page_iterator:
        subnets.extend(page["Subnets"])

    log.info("Found %s subnets for the specified zones.", len(subnets))
    return subnets
except ClientError as err:
    log.error(
        f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidVpcID.NotFound":
        log.error(
            "The specified VPC ID does not exist. "
            "Please check the VPC ID and try again."
        )
    # Add more error-specific handling as needed
    log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client

    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
        specifies how
        the load balancer forwards requests to instances in the group and how
        instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
        times and
        lower thresholds. In production, you might want to decrease the
        sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
```

```
        HealthCheckTimeoutSeconds=5,
        HealthyThresholdCount=2,
        UnhealthyThresholdCount=2,
        VpcId=vpc_id,
    )
    target_group = response["TargetGroups"][0]
    log.info(f"Created load balancing target group
'{target_group_name}'.")
    return target_group
except ClientError as err:
    log.error(
        f"Couldn't create load balancing target group
'{target_group_name}'."
    )
    error_code = err.response["Error"]["Code"]

    if error_code == "DuplicateTargetGroupName":
        log.error(
            f"Target group name {target_group_name} already exists. "
            "Check if the target group already exists."
            "Consider using a different name or deleting the existing
target group if appropriate."
        )
    elif error_code == "TooManyTargetGroups":
        log.error(
            "Too many target groups exist in the account. "
            "Consider deleting unused target groups to create space for
new ones."
        )
    log.error(f"Full error:\n\t{err}")

def delete_target_group(self, target_group_name) -> None:
    """
    Deletes the target group.
    """
    try:
        # Describe the target group to get its ARN
        response =
self.elb_client.describe_target_groups(Names=[target_group_name])
        tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

        # Delete the target group
        self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
```

```

        log.info("Deleted load balancing target group %s.",
target_group_name)

        # Use a custom waiter to wait until the target group is no longer
available
        self.wait_for_target_group_deletion(self.elb_client, tg_arn)
        log.info("Target group %s successfully deleted.", target_group_name)

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to delete target group '{target_group_name}'.")
        if error_code == "TargetGroupNotFound":
            log.error(
                "Load balancer target group either already deleted or never
existed. "
                "Verify the name and check that the resource exists in the
AWS Console."
            )
        elif error_code == "ResourceInUseException":
            log.error(
                "Target group still in use by another resource. "
                "Ensure that the target group is no longer associated with
any load balancers or resources.",
            )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                )
            return

```

```
        else:
            raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )

    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
and forwards requests to the specified target group.

:param load_balancer_name: The name of the load balancer to create.
:param subnet_ids: A list of subnets to associate with the load balancer.
:return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )
```

```

        if error_code == "DuplicateLoadBalancerNameException":
            log.error(
                f"A load balancer with the name '{load_balancer_name}'
already exists. "
                "Load balancer names must be unique within the AWS region. "
                "Please choose a different name and try again."
            )
        if error_code == "TooManyLoadBalancersException":
            log.error(
                "The maximum number of load balancers has been reached in
this account and region. "
                "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
            )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer

def create_listener(
    self,
    load_balancer_name: str,
    target_group: Dict[str, Any],
) -> Dict[str, Any]:
    """
    Creates a listener for the specified load balancer that forwards requests
to the
    specified target group.

    :param load_balancer_name: The name of the load balancer to create a
listener for.
    :param target_group: An existing target group that is added as a listener
to the
                           load balancer.
    :return: Data about the newly created listener.
    """
    try:
        # Retrieve the load balancer ARN
        load_balancer_response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        load_balancer_arn = load_balancer_response["LoadBalancers"][0][
            "LoadBalancerArn"

```

```
]

# Create the listener
response = self.elb_client.create_listener(
    LoadBalancerArn=load_balancer_arn,
    Protocol=target_group["Protocol"],
    Port=target_group["Port"],
    DefaultActions=[
        {
            "Type": "forward",
            "TargetGroupArn": target_group["TargetGroupArn"],
        }
    ],
)
log.info(
    f"Created listener to forward traffic from load balancer
    '{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
)
return response["Listeners"][0]
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to add a listener on '{load_balancer_name}' for target
        group '{target_group['TargetGroupName']}'."
    )

    if error_code == "ListenerNotFoundException":
        log.error(
            f"The listener could not be found for the load balancer
            '{load_balancer_name}'. "
            "Please check the load balancer name and target group
            configuration."
        )
    if error_code == "InvalidConfigurationRequestException":
        log.error(
            f"The configuration provided for the listener on load
            balancer '{load_balancer_name}' is invalid. "
            "Please review the provided protocol, port, and target group
            settings."
        )
    log.error(f"Full error:\n\t{err}")

def delete_load_balancer(self, load_balancer_name) -> None:
```

```
"""
Deletes a load balancer.

:param load_balancer_name: The name of the load balancer to delete.
"""
try:
    response = self.elb_client.describe_load_balancers(
        Names=[load_balancer_name]
    )
    lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
    self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
    log.info("Deleted load balancer %s.", load_balancer_name)
    waiter = self.elb_client.get_waiter("load_balancers_deleted")
    log.info("Waiting for load balancer to be deleted...")
    waiter.wait(Names=[load_balancer_name])
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
    )

    if error_code == "LoadBalancerNotFoundException":
        log.error(
            f"The load balancer '{load_balancer_name}' does not exist. "
            "Please check the name and try again."
        )
    log.error(f"Full error:\n\t{err}")

def get_endpoint(self, load_balancer_name) -> str:
    """
    Gets the HTTP endpoint of the load balancer.

    :return: The endpoint.
    """
    try:
        response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        return response["LoadBalancers"][0]["DNSName"]
    except ClientError as err:
        log.error(
```

```

        f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "LoadBalancerNotFoundException":
        log.error(
            "Verify load balancer name and ensure it exists in the AWS
console."
        )
    log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:

```

```

"""
Checks the health of the instances in the target group.

:return: The health status of the target group.
"""
try:
    tg_response = self.elb_client.describe_target_groups(
        Names=[target_group_name]
    )
    health_response = self.elb_client.describe_target_health(
        TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
    )
except ClientError as err:
    log.error(f"Couldn't check health of {target_group_name} target(s).")
    error_code = err.response["Error"]["Code"]
    if error_code == "LoadBalancerNotFoundException":
        log.error(
            "Load balancer associated with the target group was not
found. "
            "Ensure the load balancer exists, is in the correct AWS
region, and "
            "that you have the necessary permissions to access it.",
        )
    elif error_code == "TargetGroupNotFoundException":
        log.error(
            "Target group was not found. "
            "Verify the target group name, check that it exists in the
correct region, "
            "and ensure it has not been deleted or created in a different
account.",
        )
    log.error(f"Full error:\n\t{err}")
else:
    return health_response["TargetHealthDescriptions"]

```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
```

```
"""
Encapsulates a DynamoDB table to use as a service that recommends books,
movies,
and songs.
"""

def __init__(self, table_name: str, dynamodb_client: boto3.client):
    """
    Initializes the RecommendationService class with the necessary
    parameters.

    :param table_name: The name of the DynamoDB recommendations table.
    :param dynamodb_client: A Boto3 DynamoDB client.
    """
    self.table_name = table_name
    self.dynamodb_client = dynamodb_client

def create(self) -> Dict[str, Any]:
    """
    Creates a DynamoDB table to use as a recommendation service. The table
    has a
    hash key named 'MediaType' that defines the type of media recommended,
    such as
    Book or Movie, and a range key named 'ItemId' that, combined with the
    MediaType,
    forms a unique identifier for the recommended item.

    :return: Data about the newly created table.
    :raises RecommendationServiceError: If the table creation fails.
    """
    try:
        response = self.dynamodb_client.create_table(
            TableName=self.table_name,
            AttributeDefinitions=[
                {"AttributeName": "MediaType", "AttributeType": "S"},
                {"AttributeName": "ItemId", "AttributeType": "N"},
            ],
            KeySchema=[
                {"AttributeName": "MediaType", "KeyType": "HASH"},
                {"AttributeName": "ItemId", "KeyType": "RANGE"},
            ],
            ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
        )
```

```

        log.info("Creating table %s...", self.table_name)
        waiter = self.dynamodb_client.get_waiter("table_exists")
        waiter.wait(TableName=self.table_name)
        log.info("Table %s created.", self.table_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceInUseException":
            log.info("Table %s exists, nothing to be done.", self.table_name)
        else:
            raise RecommendationServiceError(
                self.table_name, f"ClientError when creating table: {err}."
            )
    else:
        return response

def populate(self, data_file: str) -> None:
    """
    Populates the recommendations table from a JSON file.

    :param data_file: The path to the data file.
    :raises RecommendationServiceError: If the table population fails.
    """
    try:
        with open(data_file) as data:
            items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
    except ClientError as err:
        raise RecommendationServiceError(
            self.table_name, f"Couldn't populate table from {data_file}:
{err}"
        )

def destroy(self) -> None:
    """
    Deletes the recommendations table.

    :raises RecommendationServiceError: If the table deletion fails.
    """
    try:

```

```

        self.dynamodb_client.delete_table(TableName=self.table_name)
        log.info("Deleting table %s...", self.table_name)
        waiter = self.dynamodb_client.get_waiter("table_not_exists")
        waiter.wait(TableName=self.table_name)
        log.info("Table %s deleted.", self.table_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            log.info("Table %s does not exist, nothing to do.",
self.table_name)
        else:
            raise RecommendationServiceError(
                self.table_name, f"ClientError when deleting table: {err}."
            )

```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```

class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
    to drive
    the demonstration of resilient architecture, such as failure of a dependency
    or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
recommendation
                        service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

```

```

def reset(self) -> None:
    """
    Resets the Systems Manager parameters to starting values for the demo.
    These are the name of the DynamoDB recommendation table, no response when
a
dependency fails, and shallow health checks.
    """
    self.put(self.table, self.table_name)
    self.put(self.failure_response, "none")
    self.put(self.health_check, "shallow")

def put(self, name: str, value: str) -> None:
    """
    Sets the value of a named Systems Manager parameter.

    :param name: The name of the parameter.
    :param value: The new value of the parameter.
    :raises ParameterHelperError: If the parameter value cannot be set.
    """
    try:
        self.ssm_client.put_parameter(
            Name=name, Value=value, Overwrite=True, Type="String"
        )
        log.info("Setting parameter %s to '%s'.", name, value)
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to set parameter {name}.")
        if error_code == "ParameterLimitExceeded":
            log.error(
                "The parameter limit has been exceeded. "
                "Consider deleting unused parameters or request a limit
increase."
            )
        elif error_code == "ParameterAlreadyExists":
            log.error(
                "The parameter already exists and overwrite is set to False.
"
                "Use Overwrite=True to update the parameter."
            )
        log.error(f"Full error:\n\t{err}")

```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAvailabilityZones](#)
 - [DescribeIamInstanceProfileAssociations](#)
 - [DescribeInstances](#)
 - [DescribeLoadBalancers](#)
 - [DescribeSubnets](#)
 - [DescribeTargetGroups](#)
 - [DescribeTargetHealth](#)
 - [DescribeVpcs](#)
 - [RebootInstances](#)
 - [ReplacelamInstanceProfileAssociation](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden Sie diesen Service mit einem SDK AWS](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Probleme in Amazon EC2 Auto Scaling beheben

Amazon EC2 Auto Scaling liefert genaue Fehlermeldungen mit einer kurzen Beschreibung, damit Sie Probleme leichter beheben können. Sie finden die Fehlermeldungen in der Beschreibung der Skalierungen.

Themen

- [Abrufen einer Fehlermeldung aus Skalierungen](#)
- [Schalten Sie Skalierungsaktivitäten aus](#)
- [Weitere Ressourcen zur Fehlerbehebung](#)
- [Fehlersuche bei Amazon EC2 Auto Scaling: Startfehler von EC2-Instance](#)
- [Fehlersuche bei Amazon EC2 Auto Scaling: AMI-Probleme](#)
- [Fehlersuche bei Amazon EC2 Auto Scaling: Load Balancer-Probleme](#)
- [Fehlersuche bei Amazon EC2 Auto Scaling: Startvorlagen](#)

Abrufen einer Fehlermeldung aus Skalierungen

Wenn Sie eine Fehlermeldung aus der Beschreibung der Skalierungsaktivitäten abrufen möchten, verwenden Sie den Befehl [describe-scaling-activities](#). Sie haben eine Aufzeichnung von Skalierungsaktivitäten, die sechs Wochen zurückreicht. Skalierungsaktivitäten werden nach Startzeit sortiert, wobei die neuesten Skalierungsaktivitäten zuerst aufgelistet werden.

Note

Die Skalierungen werden auch im Aktivitätsverlauf in der Amazon EC2 Auto Scaling-Konsole auf der Registerkarte Activity (Aktivität) für die Auto-Scaling-Gruppe angezeigt.

Verwenden Sie den folgenden Befehl, um die Skalierungsaktivitäten für eine bestimmte Auto-Scaling-Gruppe anzuzeigen.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

Im Folgenden sehen Sie eine Beispielantwort, in der der aktuelle Status der Aktivität unter `StatusCode` und die Fehlermeldung unter `StatusMessage` zu finden ist.

```
{
  "Activities": [
    {
      "ActivityId": "3b05dbf6-037c-b92f-133f-38275269dc0f",
      "AutoScalingGroupName": "my-asg",
      "Description": "Launching a new EC2 instance: i-003a5b3ffe1e9358e. Status Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle Action Completed with ABANDON Result",
      "Cause": "At 2021-01-11T00:35:52Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2021-01-11T00:35:53Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.",
      "StartTime": "2021-01-11T00:35:55.542Z",
      "EndTime": "2021-01-11T01:06:31Z",
      "StatusCode": "Cancelled",
      "StatusMessage": "Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle Action Completed with ABANDON Result",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2b\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:283179a2-f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
    ...
  ]
}
```

Eine Beschreibung der Felder in der Ausgabe finden Sie unter [Aktivität](#) in der Amazon EC2 Auto Scaling API-Referenz.

Anzeigen von Skalierungsaktivitäten für eine gelöschte-Gruppe

Um Skalierungsaktivitäten nach dem Löschen der Auto Scaling Scaling-Gruppe anzuzeigen, fügen Sie dem [describe-scaling-activities](#) Befehl die `--include-deleted-groups` Option wie folgt hinzu.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg --include-deleted-groups
```

Nachfolgend finden Sie eine Beispielantwort mit einer Skalierungsaktivität für eine gelöschte Gruppe.

```
{
  "Activities": [
    {
      "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
      "AutoScalingGroupName": "my-asg",
      "Description": "Launching a new EC2 instance. Status Reason: Your Spot request price of 0.001 is lower than the minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance failed.",
      "Cause": "At 2021-01-13T20:47:24Z a user request update of AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 0 to 3. At 2021-01-13T20:47:27Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 3.",
      "StartTime": "2021-01-13T20:47:30.094Z",
      "EndTime": "2021-01-13T20:47:30Z",
      "StatusCode": "Failed",
      "StatusMessage": "Your Spot request price of 0.001 is lower than the minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance failed.",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2b\"...}",
      "AutoScalingGroupState": "Deleted",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:283179a2-f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
    ...
  ]
}
```

Schalten Sie Skalierungsaktivitäten aus

Sie haben die folgenden Optionen, wenn Sie ein Problem untersuchen möchten, ohne dass es zu Störungen durch Skalierungsrichtlinien oder geplante Aktionen kommt:

- Verhindern Sie, dass alle dynamischen Skalierungsrichtlinien und geplanten Aktionen Änderungen an der gewünschten Kapazität der Gruppe bewirken, indem Sie die `AlarmNotification ScheduledActions` Endprozesse aussetzen. Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#).

- Deaktivieren Sie einzelne dynamische Skalierungsrichtlinien, damit sie nicht die gewünschte Kapazität der Gruppe als Reaktion auf Laständerungen ändern. Weitere Informationen finden Sie unter [Eine Skalierungsrichtlinie für eine Auto-Scaling-Gruppe deaktivieren](#).
- Aktualisieren Sie die Skalierungsrichtlinien für die individuelle Zielverfolgung so, dass sie nur horizontal skalieren (Kapazität hinzufügen), indem Sie den Scale-In-Teil der Richtlinie deaktivieren. Diese Methode verhindert, dass die gewünschte Kapazität der Gruppe schrumpft, ermöglicht es jedoch, sie bei steigender Auslastung zu erhöhen. Weitere Informationen finden Sie unter [Skalierungsrichtlinien für die Ziel-Nachverfolgung für Amazon EC2 Auto Scaling](#).
- Aktualisieren Sie Ihre Richtlinie zur vorausschauenden Skalierung auf den Modus „Nur Prognosen“. Im Modus „Nur Prognose“ generiert die vorausschauende Skalierung zwar weiterhin Prognosen, erhöht aber nicht automatisch die Kapazität. Weitere Informationen finden Sie unter [Erstellen Sie eine Richtlinie für vorausschauende Skalierung für eine Auto Scaling Scaling-Gruppe](#).

Weitere Ressourcen zur Fehlerbehebung

Auf den folgenden Seiten finden Sie zusätzliche Informationen zur Behebung von Problemen mit Amazon EC2 Auto Scaling.

- [Eine Skalierung für eine Auto-Scaling-Gruppe überprüfen](#)
- [Überwachungsgrafiken in der Amazon EC2 Auto Scaling-Konsole anzeigen](#)
- [Zustandsprüfungen für Instances in einer Auto-Scaling-Gruppe](#)
- [Überlegungen zu und Einschränkungen für Lebenszyklus-Hooks](#)
- [Eine Lebenszyklusaktion in einer Auto Scaling Scaling-Gruppe abschließen](#)
- [Stellen Sie Netzwerkkonnektivität für Ihre Auto-Scaling-Instances mit Amazon VPC bereit](#)
- [Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe](#)
- [Eine Skalierungsrichtlinie für eine Auto-Scaling-Gruppe deaktivieren](#)
- [Amazon EC2 Auto Scaling Scaling-Prozesse aussetzen und fortsetzen](#)
- [Steuern welche Auto-Scaling-Instances beim Abskalieren beendet werden](#)
- [Löschen der Auto-Scaling-Infrastruktur](#)
- [Kontingente für Auto Scaling Scaling-Ressourcen und Gruppen](#)

Die folgenden AWS Ressourcen können ebenfalls hilfreich sein:

- [Themen zu Amazon EC2 Auto Scaling im AWS Knowledge Center](#)

- [Fragen zu Amazon EC2 Auto Scaling auf re:POST AWS](#)
- [Beiträge zu Amazon EC2 Auto Scaling im AWS Compute-Blog](#)
- [Fehlerbehebung CloudFormation im CloudFormation Benutzerhandbuch](#)

Die Fehlerbehebung erfordert oft eine iterative Abfrage und Erkennung durch einen Experten oder eine Community von Helfern. Wenn Sie nach dem Ausprobieren der Vorschläge in diesem Abschnitt weiterhin Probleme haben, wenden Sie sich an AWS Support (klicken Sie auf Support AWS-Managementkonsole, Support Center) oder stellen Sie mithilfe des Amazon EC2 Auto Scaling-Tags eine Frage zu [AWS re:POST](#).

Fehlersuche bei Amazon EC2 Auto Scaling: Startfehler von EC2-Instance

Auf dieser Seite finden Sie Informationen zu Ihren EC2-Instances, die nicht gestartet werden konnten, mögliche Ursachen und Maßnahmen, die Sie zum Lösen der Probleme ergreifen können.

Wie Sie eine Fehlermeldung abrufen, erfahren Sie unter [Abrufen einer Fehlermeldung aus Skalierungen](#).

Falls Ihre EC2-Instances nicht gestartet werden können, erscheinen eine oder mehrere der folgenden Fehlermeldungen:

Startprobleme

- [Die angefragte Konfiguration wird derzeit nicht unterstützt.](#)
- [Die Sicherheitsgruppe <Name der Sicherheitsgruppe> ist nicht vorhanden. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Das Schlüsselpaar <mit Ihrer EC2-Instance verbundenes Schlüsselpaar> ist nicht vorhanden. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Der von Ihnen angeforderte Instance-Typ \(<Instance type>\) wird in der von Ihnen angeforderten Availability Zone \(<instance Availability Zone>\) nicht unterstützt...](#)
- [Ihr Spot-Anfragepreis von 0,015 ist niedriger als der erforderliche Mindestpreis für Spot-Anfragen von 0,0735...](#)
- [Ungültiger Gerätenamenname <Gerätenamenname> / Ungültiger Gerätenamenname beim Hochladen. Die EC2-Instance konnte nicht gestartet werden.](#)

- [Der Wert \(<Name des verbundenen Instance-Speichergeräts>\) für den Parameter virtualName ist ungültig... Die EC2-Instance konnte nicht gestartet werden.](#)
- [EBS-Blockgerätozuordnungen werden für den Instance-Store nicht unterstützt. AMIs](#)
- [Platzierungsgruppen dürfen nicht mit Instanzen des Typs '<Instance type>' verwendet werden. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Kunde. InternalError: Client-Fehler beim Start.](#)
- [Wir haben derzeit nicht genügend <instance type>-Kapazität in der Availability Zone, die Sie angefragt haben. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Die angefragte Reservierung ist nicht ausreichend kompatibel und hat nicht genügend freie Kapazität für diese Anfrage. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Ihre Kapazitätsblock-Reservierung <reservation id> ist noch nicht aktiv. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Es ist keine Spot-Kapazität verfügbar, die Ihrer Anforderung entspricht. Die EC2-Instance konnte nicht gestartet werden.](#)
- [<number of instances> Instance wird/Instances werden bereits ausgeführt. Die EC2-Instance konnte nicht gestartet werden.](#)

Die angefragte Konfiguration wird derzeit nicht unterstützt.

Ursache: Einige Optionen in Ihrer Startvorlage oder Startkonfiguration sind möglicherweise nicht mit dem Instance-Typ kompatibel, oder die Instance-Konfiguration wird in der von Ihnen angeforderten AWS Region oder Availability Zones möglicherweise nicht unterstützt.

Lösung: Versuchen Sie es mit einer anderen Instanzkonfiguration. Informationen zur Suche nach einem Instance-Typ, der Ihren Anforderungen entspricht, [finden Sie unter Suchen nach einem Amazon EC2 EC2-Instance-Typ](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zum Beheben dieses Problems finden Sie unter:

- Stellen Sie sicher, dass Sie ein AMI ausgewählt haben, das von Ihrem Instance-Typ unterstützt wird. Wenn der Instance-Typ beispielsweise einen ARM-basierten AWS Graviton-Prozessor anstelle eines Intel Xeon-Prozessors verwendet, benötigen Sie ein ARM-kompatibles AMI. Weitere Informationen zur Auswahl eines kompatiblen Instance-Typs finden Sie unter [Kompatibilität bei der Änderung des Instance-Typs](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Testen Sie, ob der Instance-Typ in Ihrer angeforderten Region und Availability Zones verfügbar ist. Die Instance-Typen der neuesten Generation sind möglicherweise noch nicht in einer

bestimmten Region oder Availability Zone verfügbar. Die Instance-Typen der älteren Generation sind möglicherweise nicht in neueren Regionen oder Availability Zones verfügbar. Verwenden Sie zum Suchen nach Instance-Typen, die nach Standort (Region oder Availability Zone) angeboten werden, den [describe-instance-type-offerings](#)-Befehl. Weitere Informationen [finden Sie unter Suchen nach einem Amazon EC2 EC2-Instance-Typ](#) im Amazon EC2 EC2-Benutzerhandbuch.

- Wenn Sie Dedicated Instances oder Dedicated Hosts verwenden, stellen Sie sicher, dass Sie einen Instance-Typ ausgewählt haben, der als Dedicated Instance oder Dedicated Host unterstützt wird.

Die Sicherheitsgruppe <Name der Sicherheitsgruppe> ist nicht vorhanden.
Die EC2-Instance konnte nicht gestartet werden.

Ursache: Die Sicherheitsgruppe in Ihrer Startkonfiguration wurde möglicherweise gelöscht.

Lösung:

1. Verwenden Sie den Befehl [describe-security-groups](#), um eine Liste der Sicherheitsgruppen abzurufen, die Ihrem Konto zugeordnet sind.
2. Wählen Sie in der Liste die Sicherheitsgruppen aus, die verwendet werden sollen. Verwenden Sie den Befehl [create-security-group](#), um stattdessen eine Sicherheitsgruppe zu erstellen.
3. Erstellen Sie eine neue Startvorlage oder Startkonfiguration.
4. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Das Schlüsselpaar <mit Ihrer EC2-Instance verbundenes Schlüsselpaar> ist nicht vorhanden. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Das Schlüsselpaar, mit dem die Instance gestartet wurde, wurde möglicherweise gelöscht.

Lösung:

1. Verwenden Sie den Befehl [describe-key-pairs](#), um eine Liste der Schlüsselpaare abzurufen, die Ihnen zur Verfügung stehen.
2. Wählen Sie in der Liste das Schlüsselpaar aus, das verwendet werden soll. Verwenden Sie den Befehl [create-key-pair](#), um stattdessen ein Schlüsselpaar zu erstellen.
3. Erstellen Sie eine neue Startvorlage oder Startkonfiguration.

4. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Der von Ihnen angeforderte Instance-Typ (<Instance type>) wird in der von Ihnen angeforderten Availability Zone (<instance Availability Zone>) nicht unterstützt...

Fehlermeldung: Der von Ihnen angeforderte Instance-Typ (<instance type>) wird in der von Ihnen angeforderten Availability Zone (<Instance Availability Zone>) nicht unterstützt...(Launching EC2 instance failed)... Starten der EC2-Instance fehlgeschlagen.

Ursache: Die in Ihrer Auto Scaling-Gruppe angegebenen Availability Zones unterstützen den von Ihnen gewählten Instance-Typ nicht.

Lösung:

1. Überprüfen Sie mit dem [describe-instance-type-offerings](#) Befehl oder von der Amazon EC2 EC2-Konsole aus, welche Availability Zones den von Ihnen ausgewählten Instance-Typ unterstützen, indem Sie den Wert Availability Zones im Netzwerkbereich der Seite Instance-Typen überprüfen.
2. Aktualisieren oder entfernen Sie das Subnetz für alle nicht unterstützten Zonen in den Einstellungen Ihrer Auto Scaling Scaling-Gruppe mit dem [update-auto-scaling-group](#) Befehl. Weitere Informationen finden Sie unter [Fügen Sie eine Availability Zone hinzu](#).

Ihr Spot-Anfragepreis von 0,015 ist niedriger als der erforderliche Mindestpreis für Spot-Anfragen von 0,0735...

Ursache: Der Spot-Höchstpreis in Ihrer Anfrage ist niedriger als der Spot-Preis für den ausgewählten Instance-Typ.

Lösung: Senden Sie eine neue Anforderung mit einem höheren Spot-Höchstpreis (möglicherweise dem On-Demand-Preis). Bisher basierte der von Ihnen gezahlte Spot-Preis auf Geboten. Heute zahlen Sie den aktuellen Spot-Preis. Wenn Sie Ihren Höchstpreis höher setzen, ist die Chance höher, dass der Amazon EC2 Spot-Service Ihre erforderliche Kapazität startet und erhält.

Ungültiger Geräteiname <Geräteiname> / Ungültiger Geräteiname beim Hochladen. Die EC2-Instance konnte nicht gestartet werden.

Ursache 1: Die Blockgerät-Zuweisungen in Ihrer Startvorlage enthalten möglicherweise Blockgerätenamen, die nicht verfügbar sind oder derzeit nicht unterstützt werden.

Lösung:

1. Überprüfen Sie, welche Gerätenamen für Ihre spezielle Instance-Konfiguration verfügbar sind. Weitere Informationen zur Benennung von Geräten finden Sie unter [Gerätenamen auf Linux-Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Erstellen Sie manuell eine Amazon EC2-Instance, die nicht Teil der Auto-Scaling-Gruppe ist, und untersuchen Sie das Problem. Wenn die Konfiguration für die Benennung der Blockgeräte nicht mit den Namen im Amazon Machine Image (AMI) übereinstimmt, schlägt die Instance beim Start fehl. Weitere Informationen finden Sie unter [Gerätezuordnungen blockieren](#) im Amazon EC2 EC2-Benutzerhandbuch.
3. Nachdem Sie bestätigt haben, dass Ihre Instance erfolgreich gestartet wurde, verwenden Sie den Befehl [describe-volumes](#), um zu sehen, wie die Volumes der Instance ausgesetzt sind.
4. Erstellen Sie eine neue Startvorlage oder Startkonfiguration mit dem Gerätenamen, der in der Volume-Beschreibung aufgeführt ist.
5. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Der Wert (<Name des verbundenen Instance-Speichergeräts>) für den Parameter virtualName ist ungültig... Die EC2-Instance konnte nicht gestartet werden.

Ursache: Das Format, in dem der virtuelle Name des Blockgeräts angegeben wurde, ist falsch.

Lösung:

1. Erstellen Sie eine neue Startvorlage oder Startkonfiguration, indem Sie die Gerätenamen im Parameter `virtualName` angeben. Informationen zum Gerätenamenformat finden Sie unter [Gerätebenennung auf Linux-Instances](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

EBS-Blockgerätezuoordnungen werden für den Instance-Store nicht unterstützt. AMIs

Ursache: Die Blockgerät-Zuweisungen, die in der Startvorlage oder Startkonfiguration angegeben wurden, werden auf Ihrer Instance nicht unterstützt.

Lösung:

1. Erstellen Sie eine neue Startvorlage oder Startkonfiguration mit Blockgerät-Zuweisungen, die von Ihrem Instance-Typ unterstützt werden. Weitere Informationen finden Sie unter [Gerätezuweisung blockieren](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Platzierungsgruppen dürfen nicht mit Instanzen des Typs '<Instance type>' verwendet werden. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Ihre Cluster Placement-Gruppe enthält einen ungültigen Instance-Typ.

Lösung:

1. Informationen zu gültigen Instance-Typen, die von den Placement-Gruppen unterstützt werden, finden Sie unter [Placement-Gruppen](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Folgen Sie der Anleitung unter [Platzierungsgruppen](#), um eine neue Platzierungsgruppe zu erstellen.
3. Alternativ können Sie eine neue Startvorlage oder Startkonfiguration mit einem unterstützten Instance-Typ erstellen.
4. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit einer neuen Platzierungsgruppe, Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Kunde. InternalError: Client-Fehler beim Start.

Problem: Amazon EC2 Auto Scaling versucht, eine Instance zu starten, die über ein verschlüsseltes EBS-Volume verfügt, aber die serviceverknüpfte Rolle hat keinen Zugriff auf den vom AWS KMS Kunden verwalteten Schlüssel, mit dem sie verschlüsselt wurde. Weitere Informationen finden Sie unter [Erforderliche AWS KMS Schlüsselrichtlinie für die Verwendung mit verschlüsselten Volumes](#).

Ursache 1: Sie benötigen eine Schlüsselrichtlinie, welche die Berechtigung erteilt, den kundenverwalteten Schlüssel für die richtige servicebezogene Rolle zu verwenden.

Lösung 1: Erlauben Sie der serviceverknüpften Rolle, den kundenverwalteten Schlüssel wie folgt zu verwenden:

1. Ermitteln Sie, welche serviceverknüpfte Rolle für diese Auto Scaling-Gruppe verwendet werden soll.
2. Aktualisieren Sie die Schlüsselrichtlinie für den kundenverwalteten Schlüssel und erlauben Sie der serviceverknüpften Rolle, den vom Kunden verwalteten Schlüssel zu verwenden.
3. Aktualisieren Sie die Auto Scaling-Gruppe, damit sie die serviceverknüpfte Rolle verwenden kann.

Ein Beispiel für eine Schlüsselrichtlinie, mit der die serviceverknüpfte Rolle den kundenverwalteten Schlüssel verwenden kann, finden Sie unter [Beispiel 1: Schlüsselrichtlinienabschnitte, welche Zugriff auf den kundenverwalteten Schlüssel erlauben](#).

Ursache 2: Wenn sich der vom Kunden verwaltete Schlüssel und die Auto Scaling Scaling-Gruppe in unterschiedlichen AWS Konten befinden, müssen Sie den kontoübergreifenden Zugriff auf den vom Kunden verwalteten Schlüssel konfigurieren, um der richtigen serviceverknüpften Rolle die Erlaubnis zur Verwendung des vom Kunden verwalteten Schlüssels zu erteilen.

Lösung 2: Erlauben Sie der servicebezogenen Rolle im externen Konto, den kundenverwalteten Schlüssel im lokalen Konto wie folgt zu verwenden:

1. Aktualisieren Sie die Schlüsselrichtlinie für den kundenverwalteten Schlüssel, um dem Konto der Auto-Scaling-Gruppe Zugriff auf den kundenverwalteten Schlüssel zu gewähren.
2. Definieren Sie in der Auto-Scaling-Gruppe einen IAM-Benutzer oder eine IAM-Rolle zur Erstellung einer Zuwendung.
3. Ermitteln Sie, welche serviceverknüpfte Rolle für diese Auto Scaling-Gruppe verwendet werden soll.
4. Erstellen Sie eine Zuwendung für den kundenverwalteten Schlüssel, wobei die serviceverknüpfte Rolle als berechtigtes Prinzipal dient.
5. Aktualisieren Sie die Auto Scaling-Gruppe, damit sie die serviceverknüpfte Rolle verwenden kann.

Weitere Informationen finden Sie unter [Beispiel 2: Schlüsselrichtlinienabschnitte, welche Zugriff auf den kundenverwalteten Schlüssel über mehrere Konten erlauben](#).

Lösung 3: Verwenden Sie einen kundenverwalteten Schlüssel im selben AWS -Konto wie der Auto-Scaling-Gruppe

1. Kopieren Sie den Snapshot und verschlüsseln Sie ihn erneut mit einem anderen kundenverwalteten Schlüssel im Konto der Auto-Scaling-Gruppe.
2. Erlauben Sie der serviceverknüpften Rolle, den neuen kundenverwalteten Schlüssel zu verwenden. Lesen Sie die Schritte für Lösung 1.

Wir haben derzeit nicht genügend <instance type>-Kapazität in der Availability Zone, die Sie angefragt haben. Die EC2-Instance konnte nicht gestartet werden.

Error Message: Wir haben in der von Ihnen angeforderten Availability Zone (<requested Availability Zone>), derzeit nicht genügend <instance type>-Kapazität. Unser System arbeitet an der Bereitstellung zusätzlicher Kapazität. Sie können im Augenblick <instance type> Kapazität erhalten, indem Sie in Ihrer Anfrage keine Availability Zone auswählen oder <list of Availability Zones that currently supports the instance type> auswählen. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Derzeit wird die Kombination aus angefordertem Instance-Typ und Availability Zone nicht unterstützt.

Lösung: Versuchen Sie Folgendes, um das Problem zu beheben:

- Warten Sie ein paar Minuten, bis Amazon EC2 Auto Scaling Kapazitäten für diesen Instance-Typ in anderen aktivierten Availability Zones gefunden hat.
- Erweitern Sie Ihre Auto-Scaling-Gruppe auf zusätzliche Availability Zones. Weitere Informationen finden Sie unter [Fügen Sie eine Availability Zone hinzu](#).
- Befolgen Sie die bewährte Praxis, eine Vielzahl von Instance-Typen zu verwenden, damit Sie nicht von einem bestimmten Instance-Typ abhängig sind. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

Die angefragte Reservierung ist nicht ausreichend kompatibel und hat nicht genügend freie Kapazität für diese Anfrage. Die EC2-Instance konnte nicht gestartet werden.

Ursache 1: Sie haben das Limit für die Anzahl der Instances erreicht, die Sie mit einer targeted On-Demand-Kapazitätsreservierung starten können.

Lösung 1: Erhöhen Sie entweder die Anzahl der Instances, die Sie mit der targeted On-Demand-Kapazitätsreservierung starten können, oder verwenden Sie eine Kapazitätsreservierungs-Gruppe, sodass alles, was über die reservierte Kapazität hinausgeht, als reguläre On-Demand-Kapazität gestartet wird. Weitere Informationen finden Sie unter [Reservieren Sie Kapazität in bestimmten Availability Zones mit Kapazitätsreservierungen](#).

Ursache 2: Sie haben das Limit für die Anzahl der Instances erreicht, die Sie mit einem Kapazitätsblock starten können.

Bei Kapazitätsblöcken sind Sie durch die Menge der ursprünglich gekauften Kapazität eingeschränkt. Wenn die Zahl der Starts höher ist als erwartet und die gesamte verfügbare Kapazität aufgebraucht wird, führt dies dazu, dass Starts fehlschlagen. Beendete Instances durchlaufen einen langwierigen Bereinigungsprozess, bevor sie vollständig beendet werden. Während dieser Zeit können sie nicht wiederverwendet werden. Dies kann auch dazu führen, dass Starts fehlschlagen. Weitere Informationen finden Sie unter [Capacity BlocksFür Machine-Learning-Workloads verwenden](#).

Lösung 2: Versuchen Sie Folgendes, um das Problem zu beheben:

- Behalten Sie die Anfrage unverändert bei. Wenn eine Capacity Block-Instance beendet wird, müssen Sie einige Minuten warten, bis die Instance beendet ist und die Kapazität wieder verfügbar ist. Amazon EC2 Auto Scaling löst automatisch die Startanforderung aus, bis die Kapazität verfügbar ist.
- Stellen Sie sicher, dass Sie genügend Kapazität erwerben, um Ihre Spitzenauslastung bewältigen zu können, damit dieser Fehler nicht häufig auftritt.

Ihre Kapazitätsblock-Reservierung <reservation id> ist noch nicht aktiv. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Der angegebene Kapazitätsblock ist noch nicht aktiv.

Lösung: Folgen Sie dem empfohlenen Ansatz für Kapazitätsblöcke und verwenden Sie die geplante Skalierung. Auf diese Weise können Sie sicherstellen, dass Sie die gewünschte Kapazität Ihrer Auto-Scaling-Gruppe nur dann erhöhen, wenn die Reservierung aktiv ist, und sie verringern, bevor die Reservierung beendet ist.

Es ist keine Spot-Kapazität verfügbar, die Ihrer Anforderung entspricht. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Derzeit ist nicht genügend freie Kapazität vorhanden, um Ihre Anforderung nach Spot-Instances zu erfüllen.

Lösung: Versuchen Sie Folgendes, um das Problem zu beheben:

- Warten Sie einige Minuten; die Kapazität kann häufig wechseln. Amazon EC2 Auto Scaling löst automatisch die Startanforderung aus, bis die Kapazität verfügbar ist.
- Erweitern Sie Ihre Auto-Scaling-Gruppe auf zusätzliche Availability Zones. Weitere Informationen finden Sie unter [Fügen Sie eine Availability Zone hinzu](#).
- Befolgen Sie die bewährte Praxis, eine Vielzahl von Instance-Typen zu verwenden, damit Sie nicht von einem bestimmten Instance-Typ abhängig sind. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#).

<number of instances> Instance wird/Instances werden bereits ausgeführt. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Sie haben das Limit der Anzahl der Instances, die Sie in einer Region starten können, erreicht. Wenn Sie Ihr AWS Konto erstellen, legen wir Standardlimits für die Anzahl der Instances fest, die Sie pro Region ausführen können.

Lösung: Versuchen Sie Folgendes, um das Problem zu beheben:

- Wenn Ihre aktuellen Limits Ihren Bedürfnissen nicht entsprechen, können Sie eine Kontingenterhöhung auf regionaler Basis anfordern. Weitere Informationen finden Sie unter [Amazon EC2-Servicekontingente](#) im Amazon EC2 EC2-Benutzerhandbuch.
- Senden Sie eine neue Anforderung mit einer geringeren Anzahl von Instances (die Sie später erhöhen können).

Fehlersuche bei Amazon EC2 Auto Scaling: AMI-Probleme

Auf dieser Seite finden Sie Informationen zu den Problemen im Zusammenhang mit Ihren AMIs und möglichen Ursachen sowie zu den Schritten, die Sie ergreifen können, um die Probleme zu lösen.

Wie Sie eine Fehlermeldung abrufen, erfahren Sie unter [Abrufen einer Fehlermeldung aus Skalierungen](#).

Wenn Ihre EC2-Instances aufgrund von AMI-Problemen nicht gestartet werden können, erscheinen eine oder mehrere der folgenden Fehlermeldungen.

AMI-Probleme

- [Die AMI-ID <ID of your AMI> existiert nicht. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Das AMI <AMI-ID> hat den Status "Schwebend" und kann nicht ausgeführt werden. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Ungültiger Gerätenamenname <device name>. Die EC2-Instance konnte nicht gestartet werden.](#)
- [Die Architektur 'arm64' des angegebenen Instance-Typs entspricht nicht der Architektur 'x86_64' des angegebenen AMI... Das Starten der EC2-Instance ist fehlgeschlagen.](#)
- [AMI '<AMI ID>' ist deaktiviert und kann nicht ausgeführt werden. Die EC2-Instance konnte nicht gestartet werden.](#)

Important

AWS unterstützt die private gemeinsame Nutzung eines AMI mit einem anderen AWS Konto, indem die AMI-Berechtigungen geändert werden. Wenn ein AMI privat geschaltet wird, ohne gemeinsam genutzt zu werden, kann dies zu einem Autorisierungsfehler beim Starten neuer Instances führen. Weitere Informationen zum Teilen privat AMIs finden Sie unter [Teilen eines AMI mit bestimmten AWS Konten](#) im Amazon EC2 EC2-Benutzerhandbuch.

Die AMI-ID <ID of your AMI> existiert nicht. Die EC2-Instance konnte nicht gestartet werden.

- Ursache: Das AMI wurde nach dem Erstellen der Startkonfiguration möglicherweise gelöscht.
- Lösung:
 1. Erstellen Sie eine neue Startvorlage oder Startkonfiguration mit einem gültigen AMI.

2. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Das AMI <AMI-ID> hat den Status "Schwebend" und kann nicht ausgeführt werden. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Wenn Sie Ihr AMI eben erst erstellt haben (indem Sie einen Snapshot einer laufenden Instance aufgenommen oder einen anderen Weg gewählt haben), ist es möglicherweise noch nicht verfügbar.

Lösung: Sie müssen mit dem Erstellen Ihrer Startvorlage oder Startkonfiguration warten, bis Ihr AMI verfügbar ist.

Ungültiger Gerätenamen <device name>. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Wenn Sie ein EBS-Volumen an eine EC2-Instance anhängen, müssen Sie einen gültigen Gerätenamen für das Volumen angeben. Das ausgewählte AMI muss diesen Gerätenamen unterstützen.

Lösung:

1. Erstellen Sie eine neue Startvorlage oder Startkonfiguration und geben Sie den richtigen Gerätenamen für Ihr AMI an. Die empfohlene Namenskonvention variiert je nach Virtualisierungstyp des AMI. Weitere Informationen finden Sie unter [Gerätenamen](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Die Architektur 'arm64' des angegebenen Instance-Typs entspricht nicht der Architektur 'x86_64' des angegebenen AMI... Das Starten der EC2-Instance ist fehlgeschlagen.

Ursache 1: Wenn die Architektur des AMI und der in Ihrer Startvorlage oder Startkonfiguration verwendete Instance-Typ nicht übereinstimmen, erhalten Sie einen Fehler, wenn Amazon EC2 Auto Scaling versucht, eine Instance mit der inkompatiblen Instance-Konfiguration zu starten.

Lösung 1:

1. Überprüfen Sie die Architektur Ihres AMI mit dem Befehl [describe-images](#) oder von der Amazon EC2 EC2-Konsole aus, indem Sie den Wert Architecture im Detailbereich der Amazon Machine Images () AMIs -Seite überprüfen.
2. Suchen Sie mit dem [describe-instance-types](#)Befehl oder von der Amazon EC2 EC2-Konsole aus nach einem Instance-Typ, der dieselbe Architektur wie Ihr AMI hat, indem Sie die Spalte Architektur auf dem Bildschirm Instance-Typen überprüfen. Weitere Informationen zur Auswahl eines kompatiblen Instance-Typs finden Sie unter [Kompatibilität bei der Änderung des Instance-Typs](#) im Amazon EC2 EC2-Benutzerhandbuch.
3. Erstellen Sie eine neue Startvorlage oder Startkonfiguration mit einem Instance-Typ, der die gleiche Architektur wie Ihr AMI hat.
4. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Ursache 2: Amazon EC2 Auto Scaling versucht, einen Instance-Typ zu starten, der in der Richtlinie für gemischte Instances für Ihre Auto-Scaling-Gruppe angegeben ist, aber der Instance-Typ hat nicht die gleiche Architektur wie das in Ihrer Startvorlage angegebene AMI.

Lösung 1: Nehmen Sie keine Instance-Typen mit unterschiedlichen Architekturen in Ihre Richtlinie für gemischte Instances auf.

1. Überprüfen Sie die Architektur Ihres AMI mit dem Befehl [describe-images](#) oder von der Amazon EC2 EC2-Konsole aus, indem Sie den Wert Architecture im Detailbereich der Amazon Machine Images () AMIs -Seite überprüfen.
2. Überprüfen Sie die Architektur jedes Instance-Typs, den Sie in Ihre Richtlinie für gemischte Instanzen aufnehmen möchten, mithilfe des [describe-instance-types](#)Befehls oder von der Amazon EC2 EC2-Konsole aus, indem Sie die Spalte Architektur auf dem Bildschirm Instance-Typen überprüfen. Weitere Informationen zur Auswahl kompatibler Instance-Typen finden Sie unter [Kompatibilität bei der Änderung des Instance-Typs](#) im Amazon EC2 EC2-Benutzerhandbuch.
3. Aktualisieren oder entfernen Sie die inkompatiblen Instance-Typen mit dem [update-auto-scaling-group](#)Befehl aus Ihrer Auto Scaling Scaling-Gruppe.

Lösung 2: Um sowohl Arm- (Graviton2) als auch x86_64- (Intel) Instances in derselben Auto-Scaling-Gruppe zu starten, müssen Sie Startvorlagen verwenden, die von einem Arm-kompatiblen AMI bzw.

einem Intel-x86-kompatiblen AMI unterstützt werden, um den Instance-Typen in Ihrer Richtlinie für gemischte Instances zu entsprechen.

1. Überprüfen Sie die Architektur des AMI in Ihrer vorhandenen Startvorlage mit dem Befehl [describe-images](#) oder von der Amazon EC2 EC2-Konsole aus, indem Sie den Wert Architektur im Detailbereich der Seite Amazon Machine Images () AMIs überprüfen.
2. Erstellen Sie eine neue Startvorlage mit einem AMI, das der anderen Architektur entspricht, die Sie verwenden möchten.
3. Aktualisieren Sie Ihre Auto Scaling Scaling-Gruppe, um die bestehende Startvorlage zu überschreiben, und geben Sie mithilfe des [update-auto-scaling-group](#)Befehls die neue Startvorlage für jeden kompatiblen Instance-Typ an. Weitere Informationen finden Sie unter [Verwenden von mehreren Startvorlagen](#).

AMI '<AMI ID>' ist deaktiviert und kann nicht ausgeführt werden. Die EC2-Instance konnte nicht gestartet werden.

Ursache: Sie versuchen, Instances von einem AMI aus zu starten, das deaktiviert wurde. Weitere Informationen finden Sie unter [Deaktivieren eines AMI](#) im Benutzerhandbuch für Amazon EC2.

Lösung:

1. Erstellen Sie eine neue Startvorlage oder Startkonfiguration und geben Sie ein AMI an, das nicht deaktiviert ist.
2. Aktualisieren Sie Ihre Auto-Scaling-Gruppe mit der neuen Startvorlage oder Startkonfiguration mithilfe des Befehls [update-auto-scaling-group](#).

Fehlersuche bei Amazon EC2 Auto Scaling: Load Balancer-Probleme

Auf dieser Seite finden Sie Informationen zu Problemen, die vom Load Balancer Ihrer Auto-Scaling-Gruppe verursacht wurden, mögliche Ursachen und Maßnahmen, die Sie zum Lösen der Probleme ergreifen können.

Wie Sie eine Fehlermeldung abrufen, erfahren Sie unter [Abrufen einer Fehlermeldung aus Skalierungen](#).

Wenn Ihre EC2-Instances aufgrund von Problemen mit dem Load Balancer Ihrer Auto-Scaling-Gruppe nicht gestartet werden können, erscheinen eine oder mehrere der folgenden Fehlermeldungen.

Load Balancer-Probleme

- [Eine oder mehrere Zielgruppen. Das Validieren der Load Balancer-Konfiguration ist fehlgeschlagen.](#)
- [Load Balancer kann nicht gefunden <your load balancer>werden. Das Validieren der Load Balancer-Konfiguration ist fehlgeschlagen.](#)
- [Es ist kein AKTIVER Load Balancer namens <Load Balancer-Name> vorhanden. Das Aktualisieren der Load Balancer-Konfiguration ist fehlgeschlagen.](#)
- [Die EC2-Instance <instance ID> ist nicht in der VPC. Das Aktualisieren der Load Balancer-Konfiguration ist fehlgeschlagen.](#)

Note

Sie können Reachability Analyzer verwenden, um Verbindungsprobleme zu beheben, indem Sie überprüfen, ob Instances in Ihrer Auto-Scaling-Gruppe über den Load Balancer erreichbar sind. Weitere Informationen zu den verschiedenen Netzwerk-Fehlkonfigurationsproblemen, die von Reachability Analyzer automatisch erkannt werden, finden Sie unter [Reachability Analyzer – Erläuterungscodes](#) im Benutzerhandbuch zu Reachability Analyzer.

Eine oder mehrere Zielgruppen. Das Validieren der Load Balancer-Konfiguration ist fehlgeschlagen.

Problem: Wenn Ihre Auto Scaling-Gruppe Instances startet, versucht Amazon EC2 Auto Scaling zu überprüfen, ob die Elastic Load Balancing-Ressourcen, die der Auto Scaling-Gruppe zugeordnet sind, vorhanden sind. Wenn eine Zielgruppe nicht gefunden werden kann, schlägt die Skalierungsaktivität fehl und Sie erhalten den Fehler `One or more target groups not found. Validating load balancer configuration failed..`

Ursache 1: Eine an Ihre Auto Scaling-Gruppe angehängte Zielgruppe wurde gelöscht.

Lösung 1: Sie können entweder eine neue Auto Scaling-Gruppe ohne die Zielgruppe erstellen oder die ungenutzte Zielgruppe mithilfe der Amazon EC2 Auto Scaling-Konsole oder des Befehls [detach-load-balancer-target-groups](#) aus der Auto Scaling-Gruppe entfernen.

Ursache 2: Die Zielgruppe existiert, aber es gab ein Problem beim Versuch, den Zielgruppe n-ARN beim Erstellen der Auto Scaling-Gruppe anzugeben. Ressourcen werden nicht in der richtigen Reihenfolge erstellt.

Lösung 2: Erstellen Sie eine neue Auto-Scaling-Gruppe und geben Sie den Namen des Load Balancers am Ende ein.

Load Balancer kann nicht gefunden <your load balancer>werden. Das Validieren der Load Balancer-Konfiguration ist fehlgeschlagen.

Problem: Wenn Ihre Auto -Gruppe Instances startet, versucht Amazon EC2 Auto Scaling zu überprüfen, ob die Elastic Load Balancing-Ressourcen, die der Auto Scaling-Gruppe zugeordnet sind, vorhanden sind. Wenn ein Classic Load Balancer nicht gefunden werden kann, schlägt die Skalierungsaktivität fehl und Sie erhalten den Fehler `Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed..`

Ursache 1: Der Classic Load Balancer wurde gelöscht.

Lösung 1: Sie können entweder eine neue Auto Scaling Scaling-Gruppe ohne den Load Balancer erstellen oder den ungenutzten Load Balancer aus der Auto Scaling Scaling-Gruppe entfernen, indem Sie die Amazon EC2 Auto Scaling Scaling-Konsole oder den Befehl verwenden. [detach-load-balancers](#)

Ursache 2: Der Classic Load Balancer existiert, aber es gab ein Problem bei der Angabe des Load Balancer-Namens bei der Erstellung der Auto Scaling-Gruppe. Ressourcen werden nicht in der richtigen Reihenfolge erstellt.

Lösung 2: Erstellen Sie eine neue Auto-Scaling-Gruppe und geben Sie den Namen des Load Balancers am Ende ein.

Es ist kein AKTIVER Load Balancer namens <Load Balancer-Name> vorhanden. Das Aktualisieren der Load Balancer-Konfiguration ist fehlgeschlagen.

Ursache: Der angegebene Load Balancer wurde möglicherweise gelöscht.

Lösung: Sie können entweder einen neuen Load Balancer und dann eine neue Auto-Scaling-Gruppe erstellen oder eine neue Auto-Scaling-Gruppe ohne den Load Balancer einrichten.

Die EC2-Instance <instance ID> ist nicht in der VPC. Das Aktualisieren der Load Balancer-Konfiguration ist fehlgeschlagen.

Ursache: Die angegebene Instance befindet sich nicht in der VPC.

Lösung: Sie können entweder den Load Balancer der Instance löschen oder eine neue Auto-Scaling-Gruppe erstellen.

Fehlersuche bei Amazon EC2 Auto Scaling: Startvorlagen

Verwenden Sie die folgenden Informationen, um häufige Probleme zu diagnostizieren und zu beheben, die beim Erstellversuch einer Startvorlage für Ihre Auto-Scaling-Gruppe auftreten könnten.

Instances können nicht gestartet werden

Wenn Sie keine Instances mit einer bereits angegebenen Startvorlage starten können, überprüfen Sie die folgenden Hinweise zur allgemeinen Problembehandlung: [Fehlersuche bei Amazon EC2 Auto Scaling: Startfehler von EC2-Instance](#).

Sie müssen eine gültige, vollständig formatierte Startvorlage verwenden (ungültiger Wert)

Problem: Wenn Sie versuchen, eine Startvorlage für eine Auto-Scaling-Gruppe anzugeben, erhalten Sie den `You must use a valid fully-formed launch template`-Fehler. Möglicherweise tritt dieser Fehler auf, da die Werte in der Startvorlage nur überprüft werden, wenn eine Auto-Scaling-Gruppe erstellt oder aktualisiert wird, welche die Startvorlage verwendet.

Ursache 1: Wenn der `You must use a valid fully-formed launch template`-Fehler auftritt, dann gibt es Probleme, die Amazon EC2 Auto Scaling veranlassen, etwas über die Startvorlage als ungültig zu betrachten. Das ist ein generischer Fehler, der verschiedene Ursachen haben kann.

Lösung 1: Versuchen Sie die folgenden Schritte zur Fehlerbehebung:

1. Beachten Sie den zweiten Teil der Fehlermeldung, um weitere Informationen zu erhalten. Im Anschluss an den `You must use a valid fully-formed launch template`-Fehler finden Sie eine spezifischere Fehlermeldung, die das Problem identifiziert, das Sie beheben müssen.

2. Wenn Sie die Ursache nicht finden können, testen Sie Ihre Startvorlage mit dem [run-instances](#)-Befehl. Nutzen Sie die Option `--dry-run` wie im folgenden Beispiel. So können Sie das Problem reproduzieren und Einblicke in seine Ursache erhalten.

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --dry-run
```

3. Wenn ein Wert nicht gültig ist, stellen Sie sicher, dass die angegebene Ressource vorhanden ist und dass sie korrekt ist. Wenn Sie beispielsweise ein Amazon EC2-Schlüsselpaar bestimmen, muss die Ressource im Konto und in der Region vorhanden sein, in der Sie die Auto-Scaling-Gruppe erstellen oder aktualisieren.
4. Wenn erwartete Informationen fehlen, überprüfen Sie Ihre Einstellungen und passen Sie die Startvorlage nach Bedarf an.
5. Nachdem Sie Ihre Änderungen vorgenommen haben, führen Sie den [run-instances](#)-Befehl mit der `--dry-run`-Option aus, um zu überprüfen, ob Ihre Startvorlage gültige Werte verwendet.

Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe](#).

Sie sind nicht berechtigt, die Startvorlage zu verwenden (unzureichende Berechtigungen)

Problem: Wenn Sie versuchen, eine Startvorlage für eine Auto-Scaling-Gruppe anzugeben, erhalten Sie den `You are not authorized to use launch template`-Fehler.

Ursache 1: Wenn Sie versuchen, eine Startvorlage zu verwenden und die von Ihnen verwendeten IAM-Anmeldeinformationen nicht über ausreichende Berechtigungen verfügen, wird in einer Fehlermeldung darauf hingewiesen, dass Sie nicht zur Verwendung der Startvorlage berechtigt sind.

Lösung 1: Um das Problem zu lösen, versuchen Sie Folgendes:

- Vergewissern Sie sich, dass die IAM-Anmeldeinformationen, die Sie für die Anfrage verwenden, über die Berechtigung zum Aufrufen der benötigten EC2-API-Aktionen verfügen, einschließlich der Aktion `ec2:RunInstances`. Wenn Sie Tags in Ihrer Startvorlage angegeben haben, müssen Sie auch über die Berechtigung verfügen, die `ec2:CreateTags`-Aktion zu verwenden.
- Alternativ können Sie auch überprüfen, ob den IAM-Anmeldeinformationen, die Sie für die Anfrage verwenden, die `AmazonEC2FullAccess`-Richtlinie zugewiesen ist. Diese AWS verwaltete Richtlinie gewährt vollen Zugriff auf alle Amazon EC2-Ressourcen und zugehörigen Services, einschließlich Amazon EC2 Auto Scaling und Elastic Load Balancing. CloudWatch

Weitere Informationen zu den für die Verwendung von Startvorlagen erforderlichen Berechtigungen, einschließlich beispielhafter IAM-Richtlinien, finden Sie unter [Steuern des Zugriffs auf Startvorlagen mit IAM-Berechtigungen](#) im Amazon EC2 EC2-Benutzerhandbuch. Weitere Beispiele für IAM-Richtlinien finden Sie unter [Steuern Sie die Verwendung von Amazon EC2 EC2-Startvorlagen in Auto Scaling Scaling-Gruppen](#).

Ursache 2: Wenn Sie versuchen, eine Startvorlage zu verwenden, die ein Instance-Profil angibt, müssen Sie über die IAM-Berechtigung verfügen, die dem Instance-Profil zugeordnete IAM-Rolle zu übergeben.

Lösung 2: Stellen Sie sicher, dass die IAM-Anmeldeinformationen, die Sie zum Erstellen der Anforderung verwenden, über die korrekte `iam:PassRole`-Berechtigung verfügt, um die angegebene Rolle an den Amazon-EC2-Auto-Scaling-Service zu übergeben. Weitere Informationen und eine IAM-Beispielrichtlinie finden Sie unter [IAM-Rollen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden](#). Weitere Themen zur Problembehandlung im Zusammenhang mit Instance-Profilen finden Sie unter [Problembehandlung bei Amazon EC2 und IAM](#) im IAM-Benutzerhandbuch.

Ursache 3: Wenn Sie versuchen, eine Startvorlage zu verwenden, die ein AMI in einem anderen angibt AWS-Konto, und das AMI privat ist und nicht mit dem von AWS-Konto Ihnen verwendeten geteilt wird, erhalten Sie die Fehlermeldung, dass Sie nicht berechtigt sind, die Startvorlage zu verwenden.

Lösung 3: Stellen Sie sicher, dass die Berechtigungen für das AMI das von Ihnen verwendete Konto beinhalten. Weitere Informationen finden Sie unter [Ein AMI mit bestimmten Personen teilen AWS-Konten](#) im Amazon EC2 EC2-Benutzerhandbuch.

Ähnliche Informationen

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit diesem Service.

Ressource	Description
Amazon EC2 Auto Scaling API-Referenz	Die Dokumentation für jeden API-Vorgang zeigt die Anforderungsparameter und die XML-Antwort und enthält Links zu sprachspezifischen SDK-Referenzthemen.
Auto Scaling in der AWS CLI -Befehlsreferenz	Beschreibungen der AWS CLI Befehle, die Sie für die Arbeit mit Auto Scaling Scaling-Gruppen verwenden können.
AWS -Tools für PowerShell Cmdlet-Referenz	Mit den AWS Tools für PowerShell können Sie über die PowerShell Befehlszeile Skripts für Operationen auf Ihren AWS Ressourcen erstellen.
Erstellen Sie Auto Scaling Scaling-Gruppen mit AWS CloudFormation	Mit der AWS::AutoScaling::AutoScaling können Sie Ihre Auto Scaling Scaling-Gruppen ohne manuelle Aktionen erstellen, modellieren und verwalten.
Endpunkte und Kontingente für Amazon EC2 Auto Scaling in Allgemeine AWS-Referenz	Informationen zu Amazon EC2 Auto Scaling-Regionen und -Endpunkten
Produktseite	Die Hauptwebsite für Informationen zu Amazon EC2 Auto Scaling.
AWS Re:POST	AWS verwalteter Frage-und-Antwort-Service (Q & A), der von Experten geprüfte Antworten auf Ihre technischen Fragen per Crowdsourcing bietet.

Ressource	Description
Erstellen Sie ein AMI im Amazon EC2 EC2-Benutzerhandbuch	Erfahren Sie, wie sie ein Amazon Machine Image (AMI) aus Ihrer benutzerdefinierten Instance erstellen.
Stellen Sie im Amazon EC2 EC2-Benutzerhandbuch eine Connect zu Ihrer Linux-Instance her	Erfahren Sie, wie Sie eine Verbindung zu den Linux-Instances herstellen, die Sie starten.
Stellen Sie im Amazon EC2 EC2-Benutzerhandbuch eine Connect zu Ihrer Windows-Instance her	Erfahren Sie, wie Sie eine Verbindung zu den Windows-Instances herstellen, die Sie starten.
Einen Abrechnungsalarm zur Überwachung Ihrer geschätzten AWS Gebühren im CloudWatch Amazon-Benutzerhandbuch erstellen	Erfahren Sie, wie Sie Ihre geschätzten Gebühren mithilfe von überwachen können CloudWatch.
Benutzerhandbuch zum Application Auto Scaling	Erfahren Sie, wie Sie Auto Scaling für skalierbare Ressourcen für Amazon Web Services über Amazon EC2 hinaus konfigurieren.

Dokumentverlauf

In der folgenden Tabelle sind wichtige Ergänzungen zur Amazon EC2 Auto Scaling-Dokumentation ab Juli 2018 enthalten. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie den RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Sicherheits-IAM-Update	Das beinhaltet AutoScalingServiceRolePolicy jetzt die <code>ec2:CreateReplaceRootVolumeTask</code> Berechtigung, die es Ihnen ermöglicht, Root-Volumes auf Instances im Rahmen einer Instance-Aktualisierung zu ersetzen. Darüber hinaus wurde die <code>events:DescribeRule</code> Berechtigung aus der Richtlinie entfernt.	20. November 2025
Ersetzen Sie Root-Volumes	Durch den Austausch von Root-Volumes werden Ihre Instances aktualisiert, indem nur das Root-EBS-Volume ersetzt wird, während die Instance weiterhin läuft. Weitere Informationen finden Sie unter Ersetzen von Root-Volumes bei der Instance-Aktualisierung .	20. November 2025
Richtlinien für den Lebenszyklus von Instanzen	Mit Richtlinien für den Instanzlebenszyklus haben Sie die Kontrolle darüber, was passiert, wenn die Terminierungs-Lifecycle-Hooks nicht	20. November 2025

erfolgreich abgeschlossen werden. Weitere Informationen finden Sie unter [Ersetzen von Root-Volumes bei der Instance-Aktualisierung](#).

[Synchrone Bereitstellung](#)

Sie können die LaunchInstances API jetzt verwenden, um Instances mit sofortigem Feedback synchron zu starten und genau zu kontrollieren, in welcher Availability Zone Instances gestartet werden. Weitere Informationen finden Sie unter [Synchrone Bereitstellung](#).

18. November 2025

[Lambda-Benachrichtigungsziel für Lifecycle-Hooks](#)

Sie können jetzt Lifecycle-Hook-Benachrichtigungen direkt an AWS Lambda Funktionen weiterleiten. Weitere Informationen findest du unter [Benachrichtigungen AWS Lambda direkt weiterleiten](#).

28. Juli 2025

[Metriken mit hoher Auflösung](#)

Target Tracking unterstützt jetzt hochauflösende CloudWatch Metriken mit Datenpunkten auf Sekundenebene, die in kürzeren Intervallen als einer Minute veröffentlicht werden. Weitere Informationen finden Sie unter [Erstellen einer Zielverfolgungsrichtlinie mit hochauflösenden Metriken](#) für eine schnellere Reaktion.

22. November 2024

[Sicherheits-IAM-Update](#)

Die [AutoScalingServiceRolePolicy](#) verwaltete Richtlinie gewährt Resource Groups jetzt zusätzliche Berechtigungen `resource-groups:ListGroupResources`.

20. November 2024

[Leistungsschutz](#)

Wenn Sie die attributbasierte Instanztypauswahl für Ihre Auto Scaling Scaling-Gruppe verwenden, können Sie jetzt den Leistungsschutz aktivieren, um sicherzustellen, dass die ausgewählten Instance-Typen einer bestimmten Leistungsbasislinie ähnlich sind oder diese übertreffen. Weitere Informationen finden Sie unter [Erstellen einer gemischten Instanzgruppe mithilfe der attributbasierten Instanztypauswahl](#).

20. November 2024

[Präferenz für Kapazität sreservierung](#)

Sie können jetzt das Starten von Instances bei Kapazitätssreservierungen priorisieren. Weitere Informationen finden Sie unter [Kapazitätsreservierungen](#).

20. November 2024

Zonale Verschiebung	Sie können Zonal Shift jetzt verwenden, um sich nach Anwendungsbeeinträchtigungen in einer Availability Zone zu erholen. Weitere Informationen finden Sie unter Auto Scaling Group Zonal Shift .	18. November 2024
Verteilung in der Verfügbarkeitszone	Sie können jetzt eine Availability Zone-Verteilung für Ihre Auto Scaling Scaling-Gruppe auswählen. Weitere Informationen finden Sie unter Verteilung der Availability Zone für Auto Scaling Scaling-Gruppen .	7. November 2024
Sicherheits-IAM-Update	Die AutoScalingService RolePolicy verwaltete Richtlinie gewährt Amazon EC2 jetzt zusätzliche Berechtigungen (<code>ec2:GetSecurityGroupsForVpc</code> und <code>ec2:GetInstanceTypesFromInstanceRequirements</code>).	29. Februar 2024
Der Ruhezustand im warmen Pool wird zusätzlich unterstützt AWS-Regionen	Sie können jetzt Instances in einem warmen Pool in zwei weiteren Regionen in den Ruhezustand versetzen: AWS GovCloud (US-Ost) und (US-West). AWS GovCloud Weitere Informationen zu Warm-Pools finden Sie im Amazon EC2 Auto Scaling-Benutzerhandbuch unter Warm-Pools für Amazon EC2 Auto Scaling .	26. Februar 2024

[Der Ruhezustand des warmen Pools wird zusätzlich unterstützt AWS-Regionen](#)

Sie können jetzt Instances in einem warmen Pool in zwei weiteren Regionen in den Ruhezustand versetzen : Europa (Zürich) und Naher Osten (VAE). Weitere Informationen zu Warm-Pools finden Sie im Amazon EC2 Auto Scaling-Benutzerhandbuch unter [Warm-Pools für Amazon EC2 Auto Scaling](#).

21. Februar 2024

[Support für die kontoübergreifende Verwendung von Parametern](#)

Sie können jetzt einen von einem anderen gemeinsam genutzten AWS Systems Manager Parameter AWS-Konto mit Amazon EC2 Auto Scaling verwenden. Weitere Informationen finden Sie unter [Verwenden von AWS Systems Manager Parametern anstelle von AMI IDs in Startvorlagen](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

21. Februar 2024

[Neue Option zum Schutz vor Spot-Preisen](#)

Sie können jetzt Ihren Schwellenwert für den Preisschutz für Spot-Instances als Prozentsatz eines On-Demand-Preises definieren, wenn Sie die attributbasierte Instance-Typauswahl verwenden. Weitere Informationen finden Sie unter [Preisschutz](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

29. Januar 2024

[Wartungsrichtlinien für Instances](#)

Sie können jetzt eine Wartungsrichtlinie für Instances verwenden, um festzulegen, ob Instances vor oder nach der Beendigung vorhandener Instances bei Ereignissen gestartet werden, die eine Ersetzung Ihrer Instances erfordern, einschließlich einer Instance-Aktualisierung. Weitere Informationen finden Sie unter [Wartungsrichtlinie für Instances](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

15. November 2023

[Kapazitätsblöcke für ML](#)

Sie können jetzt Instances in einem Kapazitätsblock starten, indem Sie die Kapazität sblockreservierungs-ID angeben, wenn Sie eine Startvorlage erstellen. Mit Kapazitätsblöcken können Sie GPU-Instances für ein späteres Datum reservieren, um kurzfristige Workloads für Machine Learning (ML) zu unterstützen. Weitere Informationen finden Sie unter [Verwenden von Kapazität sblocken für Machine Learning](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

31. Oktober 2023

[Neue Funktionen zur Instance-Aktualisierung](#)

Sie können jetzt eine Instance-Aktualisierung so konfigurieren, dass ihr Status auf „Fehlgeschlagen“ gesetzt wird und optional ein Rollback durchgeführt wird, wenn festgestellt wird, dass ein bestimmter CloudWatch Alarm in den ALARM Status übergegangen ist. Weitere Informationen finden Sie unter [Änderungen mit einem Rollback rückgängig machen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

31. Juli 2023

Änderungen im Handbuch

Ein neues Thema über den Start von On-Demand-Instances in Kapazitätsreservierungen wurde dem Handbuch hinzugefügt. Weitere Informationen finden Sie unter [Verwenden von On-Demand-Kapazitätsreservierungen, um Kapazitäten in spezifischen Availability Zones zu reservieren](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

28. Juli 2023

Änderungen im Handbuch

Der Anleitung wurde ein neues Thema zur Migration Ihrer AWS CloudFormation Stacks von Startkonfigurationen zu Startvorlagen hinzugefügt. Weitere Informationen erhalten Sie unter [Migration von AWS CloudFormation -Stacks von Startkonfigurationen zu Startvorlagen](#) im Benutzerhandbuch zum Amazon EC2 Auto Scaling.

18. April 2023

Support für neue API-Betriebe

Diese Version fügt mit `AttachTrafficSources` , `DetachTrafficSources` und `DescribeTrafficSources` drei neue API-Operationen hinzu. Auch ein neues Feld, `TrafficSources` , wurde den Ergebnissen der `DescribeAutoScalingGroups` -Operationen hinzugefügt. Ein neues Aktivitätsstatus, `WaitingForConnectionDrainin` g , wurde den Ergebnissen der `DescribeScalingActivities` -Operationen hinzugefügt. Amazon EC2 Auto Scaling unterstützt auch `VPC_LATTICE` , einen neuen Wert für das `HealthCheckType` -Feld in `CreateAutoScalingGroup` -, `UpdateAutoScalingGroup` - und `DescribeAutoScalingGroups` - Operationen. Weitere Informationen finden Sie in der [Amazon EC2 Auto Scaling-API-Referenz](#).

31. März 2023

[Support für Amazon VPC Lattice](#)

Dies ist die allgemein verfügbare Version von VPC Lattice für Amazon EC2 Auto Scaling. Weitere Informationen finden Sie unter [Weiterleitung des Datenverkehrs zu Ihrer Auto-Scaling-Gruppe mit einer VPC-Lattice-Zielgruppe](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

31. März 2023

[Änderungen im Handbuch](#)

Der Abschnitt mit AWS CLI Beispielen für die Arbeit mit Elastic Load Balancing enthält jetzt neue und aktualisierte Beispiele. Weitere Informationen finden Sie unter [Beispiele für die Arbeit mit Elastic Load Balancing with the AWS Command Line Interface \(AWS CLI\)](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

31. März 2023

[Zusätzliche Support für vorausschauende Skalierung AWS-Regionen](#)

Sie können jetzt Richtlinien für vorausschauende Skalierung in den Regionen Naher Osten (VAE) und AWS GovCloud (USA-Ost) erstellen. Weitere Informationen finden Sie unter [Prädiktive Skalierung von Cooldowns für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

16. März 2023

[Neue Funktionen zur Instance-Aktualisierung](#)

Sie können jetzt Instances im Standby-Modus beenden oder ignorieren und Instances ersetzen oder ignorieren, die vor dem Abskalieren geschützt sind, anstatt darauf zu warten, dass sie austauschbar werden. Sie können auch Änderungen nach einer fehlgeschlagenen Instance-Aktualisierung zurücksetzen. Im Rahmen dieser Aktualisierung wurde die Dokumentation um Themen zum Zurücksetzen einer Instance-Aktualisierung, zum Abbrechen einer Instance-Aktualisierung und zum Verständnis der Standardwerte für die konfigurierbaren Parameter einer Instance-Aktualisierung erweitert. Weitere Informationen finden Sie unter [Ersetzen von Auto Scaling-Instances basierend auf einer Instance-Aktualisierung](#) im Benutzerhandbuch zum Amazon EC2 Auto Scaling.

10. Februar 2023

[Support für die Verwendung eines AWS Systems Manager Parameters für eine AMI-ID](#)

Sie können jetzt einen Systems-Manager-Parameter anstelle einer AMI-ID in Ihrer Startvorlage verwenden. Weitere Informationen finden Sie unter [Verwenden von AWS Systems Manager Parametern anstelle von AMI IDs in Startvorlagen](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

19. Januar 2023

[Empfehlungen für prädiktive Skalierung](#)

Sie können jetzt über die Konsole von Amazon EC2 Auto Scaling Empfehlungen für die Auswertung und Auswahl der richtigen Richtlinie für prädiktive Skalierung erhalten. Weitere Informationen finden Sie unter [Auswerten Ihrer Richtlinien für prädiktive Skalierung](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

18. Januar 2023

[Prädiktive Skalierung von Prognosen](#)

Die durch prädiktive Skalierung generierten Prognosen werden jetzt alle sechs Stunden statt täglich aktualisiert. Weitere Informationen finden Sie unter [Prädiktive Skalierung von Cooldowns für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

06. Januar 2023

[Support für CloudWatch metrische Mathematik](#)

Sie können jetzt Metrikerrechnungen verwenden, wenn Sie Skalierungsrichtlinien für die Zielverfolgung erstellen. Mit metrischer Mathematik können Sie mehrere CloudWatch Metriken abfragen und mathematische Ausdrücke verwenden, um neue Zeitreihen auf der Grundlage dieser Metriken zu erstellen. Weitere Informationen finden Sie unter [Erstellen einer Zielnachverfolgung](#), [s-Skalierungsrichtlinie für Amazon EC2 Auto Scaling mit Metrikerrechnungen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

08. Dezember 2022

[Aktualisieren auf Berechtigungen für serviceverknüpfte IAM-Rollen](#)

Die AutoScalingServiceRolePolicy -Richtlinie gewährt Amazon EC2 Auto Scaling jetzt zusätzliche Berechtigungen. Weitere Informationen finden Sie unter [AWS -verwaltete Richtlinien für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

6. Dezember 2022

[Neue Spot-Zuweisungsstrategie](#)

Sie können nun die preis- und kapazitätsoptimierte Zuweisungsstrategie verwenden, um Spot Instances aus den Spot-Pools anzufordern, bei denen die Wahrscheinlichkeit einer Unterbrechung am geringsten ist und die den niedrigsten Preis haben. Weitere Informationen finden Sie unter [Zuweisungsstrategien](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

10. November 2022

[Support für die prädiktive Skalierung in der Region Asien-Pazifik \(Jakarta\)](#)

Richtlinien für die prädiktive Skalierung können jetzt in der Region Asien-Pazifik (Jakarta) erstellt werden. Weitere Informationen finden Sie unter [Prädiktive Skalierung von Cooldowns für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

13. Oktober 2022

[Support benutzerdefinierte Metriken für prädiktive Skalierung in der Konsole](#)

Sie können jetzt benutzerdefinierte Metrikdaten verwenden, wenn Sie prädiktive Scaling-Richtlinien über die Amazon EC2 Auto Scaling-Konsole erstellen. Weitere Informationen finden Sie unter [Prädiktive Skalierung von Cooldowns für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

13. Oktober 2022

[CloudWatch Überwachung für prädiktive Skalierungsmetriken](#)

Sie können jetzt mithilfe von CloudWatch auf Überwachungsdaten für die prädiktive Skalierung zugreifen. CloudWatch Metrikenberechnungen nutzen, um neue Zeitreihen zu erstellen, die die Genauigkeit von Prognosedaten anzeigen. Weitere Informationen finden Sie unter [Überwachen von Predictive Scaling-Metriken mit CloudWatch](#) im Amazon EC2 Auto Scaling Benutzerhandbuch.

7. Juli 2022

[Support für die prädiktive Skalierung in der Region Asien-Pazifik \(Osaka\)](#)

Richtlinien für die prädiktive Skalierung können jetzt in der Region Asien-Pazifik (Osaka) erstellt werden. Weitere Informationen finden Sie unter [Prädiktive Skalierung von Cooldowns für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

6. Juli 2022

[Unterstützung des Warm-Pool-Ruhezustands in weiteren Regionen](#)

Instances können jetzt in vier weiteren Regionen in den Warm-Pool-Ruhezustand versetzt werden: Afrika (Kapstadt), Asien-Pazifik (Jakarta), Asien-Pazifik (Osaka) und Europa (Mailand). Weitere Informationen zu Warm-Pools finden Sie im Amazon EC2 Auto Scaling-Benutzerhandbuch unter [Warm-Pools für Amazon EC2 Auto Scaling](#).

5. Juli 2022

[Update auf Zustandsprüfungen](#)

Bei der Durchführung von Zustandsprüfungen hilft Ihnen Amazon EC2 Auto Scaling jetzt dabei, Ausfallzeiten zu minimieren, die aufgrund vorübergehender Probleme oder falsch konfigurierter Zustandsprüfungen auftreten können. Weitere Informationen finden Sie unter [Wie Amazon EC2 Auto Scaling Ausfallzeiten minimiert](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

21. Mai 2022

[Standardmäßige Instance-Vorbereitung](#)

Sie können jetzt alle Warmup- und Cooldown-Einstellungen für eine Auto Scaling-Gruppe vereinheitlichen und die Leistung von Skalierungsrichtlinien optimieren, die kontinuierlich skalieren, indem Sie das standardmäßige Aufwärmen der Instanz aktivieren. Weitere Informationen finden Sie unter [Festlegen der standardmäßigen Instance-Vorbereitung für eine Auto-Scaling-Gruppe](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

19. April 2022

[Änderungen im Handbuch](#)

Dem Leitfaden wurde ein neues Kapitel über die Integration mit anderen AWS Diensten hinzugefügt. Weitere Informationen finden Sie unter [In Amazon EC2 Auto Scaling integrierte AWS -Dienste](#) im Benutzerhandbuch zum Amazon EC2 Auto Scaling.

29. März 2022

[Aktualisieren auf Berechtigungen für serviceverknüpfte IAM-Rollen](#)

Die AutoScalingService RolePolicy -Richtlinie gewährt Amazon EC2 Auto Scaling jetzt zusätzliche Leseberechtigungen. Weitere Informationen finden Sie unter [AWS -verwaltete Richtlinien für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

28. März 2022

[Instance-Metadaten liefern den Ziellebenszyklusstatus](#)

Sie können den Ziellebenszyklusstatus einer Auto-Scaling-Instance aus den Instance-Metadaten abrufen. Weitere Informationen finden Sie unter [Abrufen des Ziellebenszyklusstatus durch Instance-Metadaten](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

24. März 2022

[Support für neue Warmpool-Funktionalität](#)

Sie können Instances jetzt in einem Warm Pool in den Ruhezustand versetzen, um Instances zu stoppen, ohne ihren Speicherinhalt (RAM) zu löschen. Sie können Instances jetzt auch beim Abskalieren in den Warm Pool zurückgeben, anstatt immer Instance-Kapazität zu beenden, die Sie später benötigen werden. Weitere Informationen finden Sie unter [Warm-Pools für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

24. Februar 2022

[Änderungen im Handbuch](#)

Die Konsole von Amazon EC2 Auto Scaling wurde mit zusätzlichen Optionen aktualisiert, die Sie beim Starten einer Instance-Aktualisierung unterstützen, bei der Überspringen aktiviert und eine gewünschte Konfiguration angegeben ist. Weitere Informationen finden Sie unter [Starten oder Abbrechen einer Instance-Aktualisierung \(Konsole\)](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

3. Februar 2022

[Benutzerdefinierte Metriken für prädiktive Skalierungsrichtlinien](#)

Sie können jetzt wählen, ob Sie benutzerdefinierte Metriken verwenden möchten, wenn Sie prädiktive Skalierungsrichtlinien erstellen. Sie können auch die Metrikmathematik verwenden, um die Metriken, die Sie in Ihre Richtlinie aufnehmen, weiter anzupassen. Weitere Informationen finden Sie unter [Erweiterte prädiktive Skalierungsrichtlinienkonfigurationen mit benutzerdefinierten Metriken](#).

24. November 2021

[Neue On-Demand-Zuweisungsstrategie](#)

Sie können jetzt wählen, ob On-Demand-Instances auf der Grundlage des Preises gestartet werden sollen (der günstigste Instance-Typ zuerst), wenn Sie eine Auto-Scaling-Gruppe erstellen, die eine Richtlinie für gemischte Instances verwendet. Weitere Informationen finden Sie unter [Zuweisungsstrategien](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

27. Oktober 2021

[Attributbasierte Auswahl des Instance-Typs](#)

Amazon EC2 Auto Scaling bietet Unterstützung für die attributbasierte Auswahl des Instance-Typs. Anstatt die Instance-Typen manuell auszuwählen, können Sie Ihre Instance-Anforderungen als eine Reihe von Attributen ausdrücken, wie z.B. vCPU, Arbeitsspeicher und Speicher. Weitere Informationen finden Sie unter [Erstellen einer Auto-Scaling-Gruppe mit attributbasierter Auswahl des Instance-Typs](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

27. Oktober 2021

[Unterstützung für das Filtern von Gruppen nach Tags](#)

Sie können Ihre Auto-Scaling-Gruppen jetzt mit Hilfe von Tag-Filtern filtern, wenn Sie mit dem Befehl `describe-auto-scaling-groups` Informationen über Ihre Auto-Scaling-Gruppen abrufen. Weitere Informationen finden Sie unter [Verwenden von Tags zum Filtern von Auto-Scaling-Gruppen](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

14. Oktober 2021

[Änderungen im Handbuch](#)

Die Amazon EC2 Auto Scaling Scaling-Konsole wurde aktualisiert, damit Sie benutzerdefinierte Kündigungsrichtlinien mit AWS Lambda erstellen können. Die Dokumentation der Konsole wurde entsprechend überarbeitet. Weitere Informationen finden Sie unter [Verwendung verschiedener Beendigungsrichtlinien \(Konsole\)](#).

14. Oktober 2021

[Support für das Kopieren von Startkonfigurationen zu Startvorlagen](#)

Sie können jetzt alle Startkonfigurationen in einer AWS Region von der Amazon EC2 Auto Scaling Scaling-Konsole aus in neue Startvorlagen kopieren.

9. August 2021

[Erweitert die Funktionalität der Instance-Aktualisierung](#)

Beim Ersetzen von Instances können Sie jetzt Aktualisierungen, z. B. eine neue Version einer Startvorlage, einschließen, indem Sie die gewünschte Konfiguration zum `start-instance-refresh`-Befehl hinzufügen. Sie können auch das Ersetzen von Instances überspringen, die bereits über die gewünschte Konfiguration verfügen, indem Sie das Überspringen des Abgleichs aktivieren. Weitere Informationen finden Sie unter [Ersetzen von Auto Scaling-Instances basierend auf einer Instance-Aktualisierung](#) im Benutzerhandbuch zum Amazon EC2 Auto Scaling.

05. August 2021

[Support für benutzerdefinierte Beendigungsrichtlinien](#)

Sie können jetzt benutzerdefinierte Kündigungsrichtlinien mit AWS Lambda erstellen. Weitere Informationen finden Sie unter [Erstellen einer benutzerdefinierten Beendigungsrichtlinie mit Lambda](#). Die Dokumentation zum Angeben von Beendigungsrichtlinien wurde entsprechend aktualisiert.

29. Juli 2021

Änderungen im Handbuch

Die Amazon EC2 Auto Scaling-Konsole wurde aktualisiert und um zusätzliche Funktionen erweitert, die Sie bei der Erstellung von geplanten Aktionen mit einer angegebenen Zeitzone unterstützen. Die Dokumentation für [Geplante Skalierung](#) wurde entsprechend überarbeitet.

3. Juni 2021

gp3-Volumes in Startkonfigurationen

Sie können jetzt gp3-Volumen in den Blockgerätezuschreibungen für Startkonfigurationen angeben.

2. Juni 2021

Support für prädiktive Skalierung

Sie können jetzt mit prädiktiver Skalierung Ihre Amazon-EC2-Auto-Scaling-Gruppen mithilfe einer Skalierungsrichtlinie proaktiv skalieren. Weitere Informationen finden Sie unter [Prädiktive Skalierung von Cooldowns für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch. Mit diesem Update beinhaltet die [AutoScalingServiceRolePolicy](#) die Richtlinie jetzt die Erlaubnis `cloudwatch:GetMetricData` API-Aktion aufzurufen.

19. Mai 2021

[Änderungen im Handbuch](#)

Sie können jetzt von auf Beispielvorgaben für Lifecycle -Hooks zugreifen GitHub. Weitere Informationen finden Sie unter [Lebenszyklus-Hooks für Amazon EC2 Auto Scaling](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

9. April 2021

[Support für Warm-Pools](#)

Sie können jetzt die Leistung (Kaltstart minimieren) und die Kosten (Überbereitstellung der Instance-Kapazität stoppen) für Anwendungen mit langen ersten Startzeiten ausgleichen, indem Sie Warm Pools zu Auto-Scaling-Gruppen hinzufügen. Weitere Informationen finden Sie unter [Warm-Pools für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

8. April 2021

[Support für Checkpoints](#)

Sie können nun Checkpoints zu einer Instance-Aktualisierung hinzufügen, um Instances in Phasen zu ersetzen und Überprüfungen für Ihre Instances an bestimmten Punkten durchzuführen. Weitere Informationen finden Sie unter [Checkpoints zu einer Instance-Aktualisierung hinzufügen](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

18. März 2021

Änderungen im Handbuch

Verbesserte Dokumentation für die Verwendung EventBridge mit Amazon EC2 Auto Scaling Scaling-Ereignissen und Lifecycle-Hooks. Weitere Informationen finden Sie unter [Verwenden von Amazon EC2 Auto Scaling mit EventBridge](#) und [Tutorial: Einen Lifecycle-Hook konfigurieren, der eine Lambda-Funktion aufruft](#) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

18. März 2021

Unterstützung für lokale Zeitzonen

Sie können jetzt wiederkehrende geplante Aktionen in der lokalen Zeitzone erstellen, indem Sie die `--time-zone` -Option dem `put-scheduled-update-group-action`-Befehl hinzufügen. Wenn Ihre Zeitzone Sommerzeit befolgt, wird die wiederkehrende Aktion automatisch für Sommerzeit angepasst. Weitere Informationen finden Sie unter [Geplante Skalierung](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

9. März 2021

[Erweitert die Funktionalität für Richtlinien für gemischte Instances](#)

Sie können jetzt Instance-Typen für Ihre Spot-Kapazität priorisieren, wenn Sie eine Richtlinie für gemischte Instances verwenden. Amazon EC2 Auto Scaling versucht, die Prioritäten des Instance-Typen auf Best Effort-Basis zu erfüllen, optimiert jedoch zuerst die Kapazität. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

8. März 2021

[Skalieren von Aktivitäten für gelöschte Gruppen](#)

Sie können jetzt Skalierungsaktivitäten für gelöschte Auto-Scaling-Gruppen anzeigen, indem Sie dem Befehl `describe-scaling-activities` die `--include-deleted-groups` Option hinzufügen. Weitere Informationen finden Sie unter [Fehlerbehebung bei Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

23. Februar 2021

[Verbesserungen an der Konsole](#)

Sie können jetzt einen Application Load Balancer oder Network Load Balancer über die Amazon EC2 Auto Scaling-Konsole erstellen und anhängen. Weitere Informationen finden Sie unter [Einen neuen Application Load Balancer oder Network Load Balancer \(Konsole\) erstellen und anhängen](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

24. November 2020

[Mehrere Netzwerkschnittstellen](#)

Sie können jetzt eine Startvorlage für eine Auto-Scaling-Gruppe konfigurieren, die mehrere Netzwerkschnittstellen angibt. Weitere Informationen finden Sie unter [Netzwerkschnittstellen in einer VPC](#).

23. November 2020

[Mehrere Startvorlagen](#)

Mehrere Startvorlagen können jetzt mit Auto-Scaling-Gruppen verwendet werden. Weitere Informationen finden Sie unter [Angeben einer anderen Einführungsvorlage für einen Instance-Typ](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

19. November 2020

[Gateway Load Balancer](#)

Aktualisierter Leitfaden, der zeigt, wie Sie einen Gateway Load Balancer an eine Auto-Scaling-Gruppe anfügen, um sicherzustellen, dass Appliance-Instances, die von Amazon EC2 Auto Scaling gestartet wurden, automatisch registriert und vom Load Balancer abgemeldet werden. Weitere Informationen finden Sie unter [Elastic Load Balancing-Typen](#) und [Anfügen eines Load Balancer an Ihre Auto-Scaling-Gruppe](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

10. November 2020

[Maximale Lebensdauer von Instances](#)

Sie können jetzt die maximale Instance-Lebensdauer auf einen Tag (86.400 Sekunden) reduzieren. Weitere Informationen finden Sie unter [Ersetzen von Auto Scaling-Instances basierend auf der maximalen Instance-Lebensdauer](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

9. November 2020

Kapazitätsausgleich

Sie können Ihre Auto-Scaling-Gruppe so konfigurieren, dass sie um eine Ersatz-Spot-Instance startet, wenn Amazon EC2 eine Empfehlung zum Neuausgleich ausgibt. Weitere Informationen dazu finden Sie unter [Amazon EC2 Auto Scaling Capacity Rebalancing](#) im Amazon EC2 Auto Scaling User Guide.

4. November 2020

Instance Metadata Service Version 2

Sie können die Verwendung des Instance-Metadaten services Version 2 verlangen. Es handelt sich um eine sitzungsorientierte Methode zum Anfordern von Instance-Metadaten, wenn Startkonfigurationen verwendet werden. Weitere Informationen finden Sie unter [Konfigurieren der Instance-Metadatenoptionen](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

28. Juli 2020

[Änderungen im Handbuch](#)

Verschiedene Verbesserungen und neue Konsolenverfahren in den Abschnitten [Steuern, welche Auto-Scaling-Instances während der Abskalierung beendet werden](#), [Überwachen der Auto-Scaling-Instances und Auto-Scaling-Gruppen](#), [Startvorlagen](#) und [Startkonfigurationen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

28. Juli 2020

[Instance-Aktualisierung](#)

Starten Sie eine Instance-Aktualisierung, um alle Instances in Ihrer Auto-Scaling-Gruppe zu aktualisieren, wenn Sie eine Konfigurationsänderung vornehmen. Weitere Informationen finden Sie unter [Ersetzen von Auto-Scaling-Instances basierend auf einer Instance-Aktualisierung](#) im Benutzerhandbuch zum Amazon EC2 Auto Scaling.

16. Juni 2020

Änderungen im Handbuch

Verschiedene Verbesserungen in den Abschnitten [Ersetzen von -Instances basierend auf der maximalen Instance-Lebensdauer](#), [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#), [Skalierung auf Grundlage von Amazon SQS](#) und [Markieren von Auto-Scaling-Gruppen und -Instances](#) im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

6. Mai 2020

Änderungen im Handbuch

Verschiedene Verbesserungen an der IAM-Dokumentation. Weitere Informationen finden Sie unter [Startvorlagen-Support](#) und [Beispiele für identitätsbasierte Amazon EC2 Auto Scaling-Richtlinien](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

4. März 2020

[Deaktivieren von Skalierungsrichtlinie](#)

Sie können jetzt Skalierungsrichtlinien deaktivieren und wieder aktivieren. Mit dieser Funktion können Sie eine Skalierungsrichtlinie vorübergehend deaktivieren, während die Konfigurationsdetails beibehalten werden, so dass Sie die Richtlinie später erneut aktivieren können. Weitere Informationen finden Sie unter [Deaktivieren einer Skalierungsrichtlinie für eine Auto-Scaling-Gruppe](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

18. Februar 2020

[Hinzufügen einer Benachrichtigungsfunktion](#)

Amazon EC2 Auto Scaling sendet jetzt Ereignisse an Sie, AWS Health Dashboard wenn Ihre Auto Scaling Scaling-Gruppen aufgrund einer fehlenden Sicherheitsgruppe oder Startvorlage nicht skalieren können. Weitere Informationen finden Sie unter [Health Dashboard -Benachrichtigungen für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

12. Februar 2020

Änderungen im Handbuch

Verschiedene Verbesserungen und Korrekturen in den Abschnitten [Funktionsweise von Amazon EC2 Auto Scaling mit IAM](#), [Identitätsbasierte Amazon EC2 Auto Scaling-Richtlinienbeispiele](#), [Erforderliche CMK-Schlüsselrichtlinie für Verwendung mit verschlüsselten Volumes](#) und [Überwachung Ihrer Auto-Scaling-Instances und -Gruppen](#) des Amazon EC2 Auto Scaling-Benutzerhandbuchs.

10. Februar 2020

Änderungen im Handbuch

Verbesserte Dokumentation für Auto-Scaling-Gruppen, die Instance-Gewichtung verwenden. Erfahren Sie, wie Sie Skalierungsrichtlinien verwenden, wenn Sie „Kapazitätseinheiten“ verwenden, um die gewünschte Kapazität zu messen. Weitere Informationen finden Sie unter [Funktionsweise von Skalierungsrichtlinien](#) und [Skalierungsanpassungstypen](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

6. Februar 2020

Neues Sicherheitskapitel

Ein neues [Sicherheits](#)-Kapitel im Benutzerhandbuch für Amazon EC2 Auto Scaling hilft Ihnen zu verstehen, wie Sie das [Modell der geteilten Verantwortung](#) anwenden, wenn Sie Amazon EC2 Auto Scaling verwenden. Als Teil dieses Updates wurde das Kapitel „Zugriffskontrolle für Ihre Amazon EC2 Auto Scaling-Ressourcen“ im Benutzerhandbuch durch den neuen, nützlicheren Abschnitt [Identity and Access Management für Amazon EC2 Auto Scaling](#) ersetzt.

4. Februar 2020

Empfehlungen für Instance-Typen

AWS Compute Optimizer bietet Amazon EC2 EC2-Instance-Empfehlungen, um Ihnen zu helfen, die Leistung zu verbessern, Geld zu sparen oder beides. Weitere Informationen finden Sie unter [Erhalten von Empfehlungen für einen Instance-Typ](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

3. Dezember 2019

[Dedicated Hosts und Hostressourcengruppen](#)

Aktualisierte Anleitung, um zu zeigen, wie eine Startvorlage erstellt wird, die eine Host-Ressourcengruppen angibt. Auf diese Weise können Sie eine Auto-Scaling-Gruppe mit einer Startvorlage erstellen, die ein BYOL-AMI angibt, die auf Dedicated Hosts verwendet wird. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Group](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

3. Dezember 2019

[Unterstützung für Amazon VPC-Endpunkte](#)

Sie können jetzt eine private Verbindung zwischen Ihrer VPC und Amazon EC2 Auto Scaling herstellen. Weitere Informationen finden Sie unter [Amazon EC2 Auto Scaling und VPC-Schnittstellenendpunkte](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

22. November 2019

[Maximale Lebensdauer von Instances](#)

Sie können Instances jetzt automatisch ersetzen, indem Sie die Maximaldauer für den Betrieb einer Instance angeben. Wenn sich Instances diesem Grenzwert nähern, werden sie von Amazon EC2 Auto Scaling schrittweise ersetzt. Weitere Informationen finden Sie unter [Ersetzen von Auto Scaling-Instances basierend auf der maximalen Instance-Lebensdauer](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

19. November 2019

[Instance-Gewichtung](#)

Für Auto-Scaling-Gruppen mit mehreren Instance-Typen können Sie nun optional die Anzahl der Kapazitätseinheiten angeben, die jeder Instance-Typ zur Kapazität der Gruppe beiträgt. Weitere Informationen finden Sie unter [Instance-Gewichtung für Amazon EC2 Auto Scaling](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

19. November 2019

[Mindestanzahl der Instance-Typen](#)

Sie müssen keine zusätzlichen Instance-Typen mehr für Gruppen von Spot, On-Demand und Reserved Instances angeben. Für alle Auto-Scaling-Gruppen beträgt der Mindestwert jetzt ein Instance-Typ. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

16. September 2019

[Unterstützung für eine neue Spot-Zuweisungsstrategie](#)

Amazon EC2 Auto Scaling unterstützt jetzt eine neue „kapazitätsoptimierte“ Spot-Zuweisungsstrategie, die Ihre Anfrage mit Spot-Instance-Pools erfüllt, die basierend auf der verfügbaren Spot-Kapazität optimal ausgewählt werden. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

12. August 2019

Änderungen im Handbuch

Verbesserte Amazon EC2 Auto Scaling-Dokumentation in den Themen [Serviceverknüpfte Rollen](#) und [Erforderliche CMK-Schlüsselrichtlinie für die Verwendung mit verschlüsselten Volumes](#).

1. August 2019

Unterstützung für Tagging-Erweiterung

Amazon EC2 Auto Scaling fügt Tags jetzt im Rahmen desselben API-Aufrufs, der die Instances startet, zu Amazon EC2 Instances hinzu. Weitere Informationen finden Sie unter [Markieren von Auto-Scaling-Gruppen und -Instances](#).

26. Juli 2019

Änderungen im Handbuch

Verbesserte Amazon EC2 Auto Scaling-Dokumentation im Thema [Aus- und Fortsetzungen von Skalierungsprozessen](#). Das Thema [Beispiele für vom Kunden verwaltete Richtlinien](#) wurde aktualisiert und enthält jetzt ein Beispiel für eine Richtlinie, die es Benutzern ermöglicht, nur bestimmte serviceverknüpfte Rollen mit benutzerdefiniertem Suffix an Amazon EC2 Auto Scaling zu übergeben.

13. Juni 2019

[Unterstützung für neue Amazon EBS-Funktion](#)

Unterstützung für neue Amazon EBS-Funktion im Startvorlagen-Thema hinzugefügt. Ändern Sie den Verschlüsselungsstatus eines EBS-Volumens während der Wiederherstellung von einem Snapshot. Weitere Informationen finden Sie unter [Erstellen einer Startvorlage für eine Auto-Scaling-Group](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

13. Mai 2019

[Änderungen im Handbuch](#)

Verbesserte Amazon EC2 Auto Scaling-Dokumentation in den folgenden Abschnitten: [Steuern der Auswahl der bei der horizontalen Skalierung nach unten zu beendenden Auto-Scaling-Instances](#), [Auto-Scaling-Gruppen](#), [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#) und [Dynamische Skalierung für Amazon EC2 Auto Scaling](#).

12. März 2019

[Unterstützung zum Kombinieren von Instance-Typen und Kaufoptionen](#)

Bereitstellung und automatische Skalierung der Instances in den Kaufoptionen (Spot-, On-Demand- und Reserved Instances) und Instance-Typen innerhalb einer einzelnen Auto-Scaling-Gruppe. Weitere Informationen finden Sie unter [Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

13. November 2018

[Thema für die Skalierung auf Basis von Amazon SQS aktualisiert](#)

Aktualisierte Anleitung mit der Beschreibung, wie Sie benutzerdefinierte Metriken verwenden, um eine Auto-Scaling-Gruppe als Reaktion auf geänderten Bedarf von einer Amazon SQS-Warteschlange zu skalieren. Weitere Informationen finden Sie unter [Skalierung basierend auf Amazon SQS](#) im Benutzerhandbuch für Amazon EC2 Auto Scaling.

26. Juli 2018

In der folgenden Tabelle werden die wichtigen Änderungen an der Amazon EC2 Auto Scaling-Dokumentation vor Juli 2018 beschrieben.

Feature	Description	Datum der Veröffentlichung
Unterstützung für Skalierungsrichtli	Richten Sie in nur wenigen Schritten eine dynamische Skalierung für Ihre Anwendung ein. Weitere Informati	12. Juli 2017

Feature	Description	Datum der Veröffentlichung
nien für die Ziel-Nachverfolgung	onen finden Sie unter Skalierungsrichtlinien für die Zielverfolgung für Amazon EC2 Auto Scaling .	
Unterstützung für Berechtigungen auf Ressourcenebene	Erstellen von IAM-Richtlinien zur Kontrolle des Zugriffs auf Ressourcenebene. Weitere Informationen finden Sie unter Steuern des Zugriffs auf Ihre Amazon EC2 Auto Scaling-Ressourcen .	15. Mai 2017
Überwachen von Verbesserungen	Auto-Scaling-Gruppenmetriken erfordern nicht mehr, dass Sie die detaillierte Überwachung aktivieren. Sie können die Erfassung von Gruppenmetriken jetzt auf der Registerkarte Monitoring der Konsole aktivieren und dort Metrikdiagramme anzeigen. Weitere Informationen finden Sie unter Überwachen Ihrer Auto Scaling Scaling-Gruppen und -Instances mithilfe von Amazon CloudWatch .	18. August 2016
Support für Application Load Balancer	Hinzufügen von Zielgruppen zu einer neuen oder bestehenden Auto-Scaling-Gruppe. Weitere Informationen finden Sie unter Anfügen eines Load Balancers an Ihre Auto-Scaling-Gruppe .	11. August 2016
Ereignisse für Lebenszyklus-Hooks	Amazon EC2 Auto Scaling sendet Ereignisse an, EventBridge wenn es Lifecycle-Hooks aufruft. Weitere Informationen finden Sie unter Ermitteln, EventBridge wann Ihre Auto Scaling-Gruppe skaliert .	24. Februar 2016
Instance-Schutz	Verhindern der Beendigung bestimmter Instances durch Amazon EC2 Auto Scaling bei der horizontalen Skalierung nach unten. Weitere Informationen finden Sie unter Instance-Schutz .	07. Dezember 2015

Feature	Description	Datum der Veröffentlichung
Richtlinien zur schrittweisen Skalierung	Erstellen einer Skalierungsrichtlinie, die Ihnen eine Skalierung auf Grundlage des Ausmaßes der Alarmüberschreitung ermöglicht. Weitere Informationen finden Sie unter Skalierungsrichtlinientypen .	06. Juli 2015
Aktualisieren des Load Balancers	Hinzufügen eines Load Balancers zu und Trennen eines Load Balancers von einer vorhandenen Auto-Scaling-Gruppe. Weitere Informationen finden Sie unter Anfügen eines Load Balancers an Ihre Auto-Scaling-Gruppe .	11. Juni 2015
Support für ClassicLink	Verknüpfen von EC2-Classic-Instances der Auto-Scaling-Gruppe mit einer VPC, was die Kommunikation zwischen diesen verknüpften EC2-Classic-Instances und den Instances in der VPC mit privaten IP-Adressen ermöglicht. Weitere Informationen finden Sie unter Verknüpfen von EC2-Classic-Instances mit einer VPC .	19. Januar 2015
Lebenszyklus-Hooks	Belassen von neu gestarteten oder in der Beendigung begriffenen Instances in einem schwebenden Status, während an ihnen Aktionen durchgeführt werden. Weitere Informationen hierzu finden Sie unter Amazon EC2 Auto Scaling-Lebenszyklus-Hooks .	30. Juli 2014
Trennen von Instances	Trennen Sie die Instances von der Auto-Scaling-Gruppe. Weitere Informationen finden Sie unter Detach EC2 Instances From Your Auto Scaling Group .	30. Juli 2014
Versetzen von Instances in einen Standby-Status	Versetzen von Instances in einem InService -Status in einen Standby-Status. Weitere Informationen finden Sie unter Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe .	30. Juli 2014

Feature	Description	Datum der Veröffentlichung
Verwalten von Tags	Verwalten Sie Ihre Auto-Scaling-Gruppen mit der AWS-Managementkonsole. Weitere Informationen finden Sie unter Markieren von Auto-Scaling-Gruppen und -Instances .	01. Mai 2014
Unterstützung für Dedicated Instances	Starten von Dedicated Instances durch Angabe eines Placement-Tenancy-Attributs beim Erstellen einer Startkonfiguration. Weitere Informationen finden Sie unter Tenancy zur Instance-Platzierung .	23. April 2014
Erstellen einer Gruppe oder Startkonfiguration aus einer EC2-Instance	Erstellen einer Auto-Scaling-Gruppe oder einer Startkonfiguration mithilfe einer EC2-Instance. Informationen zum Erstellen einer Startkonfiguration mithilfe einer EC2-Instance finden Sie unter Erstellen einer Startkonfiguration mithilfe einer EC2-Instance . Weitere Informationen zum Erstellen einer Auto-Scaling-Gruppe mithilfe einer EC2-Instance finden Sie unter Erstellen einer Auto-Scaling-Gruppe mit einer EC2-Instance .	02. Januar 2014
Hinzufügen von Instances	Aktivieren der automatischen Skalierung für eine EC2-Instance durch Hinzufügen der Instance zu einer bestehenden Auto-Scaling-Gruppe. Weitere Informationen finden Sie unter Hinzufügen von EC2-Instances zu Ihrer Auto-Scaling-Gruppe .	02. Januar 2014
Anzeigen von Kontolimits	Anzeigen der Limits von Auto-Scaling-Ressourcen für Ihr Konto. Weitere Informationen finden Sie unter Kontingente für Auto Scaling Scaling-Ressourcen und Gruppen .	02. Januar 2014
Konsolensupport für Amazon EC2 Auto Scaling	Greifen Sie mit dem auf Amazon EC2 Auto Scaling zu. AWS-Managementkonsole Weitere Informationen finden Sie unter Erste Schritte mit Amazon EC2 Auto Scaling .	10. Dezember 2013

Feature	Description	Datum der Veröffentlichung
Zuweisen einer öffentlichen IP-Adresse	Zuweisen einer öffentlichen IP-Adresse an eine Instance einer VPC. Weitere Informationen finden Sie unter Starten von Auto-Scaling-Instances in einer VPC .	19. September 2013
Instance-Beendigungsrichtlinie	Angeben einer Instance-Beendigungsrichtlinie für Amazon EC2 Auto Scaling zur Verwendung beim Beenden von EC2-Instances. Weitere Informationen finden Sie unter Steuern, welche Auto-Scaling-Instances während der horizontalen Skalierung nach unten beendet werden .	17. September 2012
Unterstützung für IAM-Rollen	Starten von EC2-Instances mit einem IAM-Instance-Profil. Sie können diese Funktion verwenden, um Instances IAM-Rollen zuzuweisen, was Anwendungen sicheren Zugriff auf andere Amazon Web Services ermöglicht. Weitere Informationen finden Sie unter Starten von Auto-Scaling-Instances mit einer IAM-Rolle .	11. Juni 2012
Unterstützung für Spot-Instances	Starten Sie Spot-Instances mit einer Startkonfiguration. Weitere Informationen finden Sie unter Anfordern von Spot-Instanzen für fehlertolerante und flexible Anwendungen .	7. Juni 2012
Markieren von Gruppen und Instances	Markieren von Auto-Scaling-Gruppen und Festlegen der Einstellung, dass das Tag auch nach seiner Erstellung gestartet EC2-Instances hinzugefügt wird. Weitere Informationen finden Sie unter Markieren von Auto-Scaling-Gruppen und -Instances .	26. Januar 2012

Feature	Description	Datum der Veröffentlichung
Support für Amazon SNS	<p>Verwenden von Amazon SNS für den Empfang von Benachrichtigungen, wenn Amazon EC2 Auto Scaling EC2-Instances startet oder beendet. Weitere Informationen finden Sie unter Erhalten von SNS-Benachrichtigungen über Skalierungen Ihrer Auto-Scaling-Gruppe.</p> <p>Amazon EC2 Auto Scaling umfasst auch die folgenden neuen Funktionen:</p> <ul style="list-style-type: none"> • Die Möglichkeit, wiederkehrende Skalierungsaktivitäten mithilfe von Cron-Syntax einzurichten. Weitere Informationen finden Sie unter PutScheduledUpdateGroupAction -API-Operation. • Eine neue Konfigurationseinstellung, mit der Sie horizontal skalieren können, ohne die gestartete Instance dem Load Balancer () LoadBalancer hinzuzufügen. Weitere Informationen finden Sie unter ProcessType -API-Datentyp. • Das Flag ForceDelete in der Operation DeleteAutoScalingGroup, das Amazon EC2 Auto Scaling zum Löschen der Auto-Scaling-Gruppe einschließlich der ihr zugeordneten Instances anweist, ohne auf die Beendigung der Instances zu warten. Weitere Informationen finden Sie unter DeleteAutoScalingGroup -API-Operation. 	20. Juli 2011
Geplante Skalierungsaktionen	Unterstützung für geplante Skalierungsaktionen hinzugefügt. Weitere Informationen finden Sie unter Geplante Skalierung für Amazon EC2 Auto Scaling .	2. Dezember 2010
Support für Amazon VPC	Support für Amazon VPC hinzugefügt. Weitere Informationen finden Sie unter Starten von Auto-Scaling-Instances in einer VPC .	2. Dezember 2010

Feature	Description	Datum der Veröffentlichung
Unterstützung für HPC-Cluster	HPC-Cluster (High Performance Computing (HPC)) werden nun unterstützt.	2. Dezember 2010
Unterstützung für Zustandsprüfungen	Die Verwendung von Elastic Load Balancing-Zustandsprüfungen mit Amazon EC2 Auto Scaling-verwalteten EC2-Instances wurde hinzugefügt. Weitere Informationen finden Sie unter Integritätsprüfungen für Instances in einer Auto Scaling Scaling-Gruppe .	2. Dezember 2010
Support für CloudWatch Alarme	Der ältere Auslösemechanismus wurde entfernt und Amazon EC2 Auto Scaling neu gestaltet, um die CloudWatch Alarmfunktion zu verwenden. Weitere Informationen finden Sie unter Dynamische Skalierung für Amazon EC2 Auto Scaling .	2. Dezember 2010
Anhalten und Fortsetzen von Skalierungen	Unterstützung für das Anhalten und Fortsetzen von Skalierungsprozessen hinzugefügt.	2. Dezember 2010
Unterstützung für IAM	IAM wird nun unterstützt. Weitere Informationen finden Sie unter Steuern des Zugriffs auf Ihre Amazon EC2 Auto Scaling-Ressourcen .	2. Dezember 2010

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.