



Hosting-Leitfaden

Amazon GameLift Servers



Amazon GameLift Servers: Hosting-Leitfaden

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Marken, die nicht im Besitz von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon GameLift Servers?	1
Schlüssel-Features	1
Vorgehensweise bei der Arbeit mit Amazon GameLift Servers	4
Optionen zum Hosten von Spielen	5
Verwaltetes EC2	5
Verwaltete Container	6
Überall, Hosting	7
Hybrides Hosting	8
Funktionsweise	9
Kernkomponenten	9
Wie Komponenten zusammenarbeiten	16
Spielererfahrung	17
Bieten Sie Spielern weltweit schnelles Gameplay	17
Hilf Spielern, schnell an Spielen teilzunehmen	18
Entwickle kreative Methoden, um Spieler für Spielsitzungen zu gruppieren	18
Sorgt während des gesamten Spiels für zuverlässige Verbindungen	19
Verbessern Sie das Spielerlebnis auf der Grundlage realer Daten	20
Integrieren Sie erweiterte Spielfunktionen in andere Dienste AWS	20
Game-Hosting in der Produktion	21
Überwachung des Zustands und der Leistung des Hostings	21
Verwaltung von Updates und Patches für Spieleserver	22
Optimierung von Leistung und Skalierung	23
Problembehebung und Live-Betrieb	23
Tools und Integration	24
Zugehörige Ressourcen	24
Wichtige Begriffe	25
Erste Schritte	33
Bevor Sie beginnen	33
Wähle deinen Weg	33
Ich möchte Amazon GameLift Servers schnell erkunden	33
Ich möchte eine maßgeschneiderte Hosting-Lösung erstellen	34
Empfohlener Lernpfad	34
Richten Sie ein AWS Konto ein	35
Melden Sie sich an für ein AWS-Konto	37

Erstellen eines Benutzers mit Administratorzugriff	37
Legen Sie Benutzerberechtigungen fest für Amazon GameLift Servers	38
Richten Sie den programmatischen Zugriff für Benutzer ein	39
Richte den programmatischen Zugriff für dein Spiel ein	42
Beispiele für IAM-Berechtigungen	42
Richten Sie eine IAM-Servicerolle ein	47
Holen Sie sich Entwicklungstools	52
Für Spieleserver	52
Für Spieleclients	55
Für Backend-Dienste für Spiele	56
Für das Ressourcenmanagement	56
Für Amazon GameLift Servers Realtime	57
Tutorial: Schnelles Onboarding mit dem Wrapper Amazon GameLift Servers	58
Voraussetzungen	59
-Übersicht	59
Schritt 1: Holen Sie sich den Gameserver-Wrapper und bauen Sie ihn	60
Schritt 2: Bereite deinen Gameserver-Build vor	60
Schritt 3: Konfigurieren Sie den Wrapper für Ihre Flotte	61
Schritt 4: Lade den Spielserver-Build hoch	63
Schritt 5: Erstellen Sie die verwaltete EC2 Flotte	63
Schritt 6: Erstelle eine Spielsitzung und stelle eine Verbindung zu ihr her	65
Schritt 7: Verwalte und überwache deine Flotte	66
Schritt 8: Skalieren Sie Ihre Spieleserver	66
Beheben Sie häufig auftretende Probleme	68
Nächste Schritte	69
Beispiele für Amazon GameLift Servers	70
Beispiel für einen benutzerdefinierten Spieleserver	70
Aufbau einer Game-Hosting-Lösung	71
Fahrpläne für die Entwicklung	72
Roadmap für die verwaltete EC2-Entwicklung	72
Entwicklungsplan für verwaltete Container	79
Roadmap für die Entwicklung von überall	86
Roadmap für die Entwicklung von Hybrid-Hosting	93
Bereite ein Spiel für das Hosten vor	101
Bereite dich mit dem Game Engine-Plugin vor	102
Integrieren Sie einen Spieleserver	193

Einen Gameserver-Build verpacken	222
Amazon GameLift ServersInteraktionen	236
Erstellen Sie einen Backend-Service	242
Authentifizieren Sie Ihre Spieler	243
Integrieren Sie die Funktionalität des Spielclients	243
Serverloses Backend	249
WebSocketbasiertes Backend	251
Stellen Sie Hosting-Flotten bereit	253
Eigenschaften der Flotte	253
Wählen Sie eine Hosting-Option	255
Verwaltete Flotten EC2	256
Verwaltete Containerflotten	269
Überall Flotten	292
Erstellen Sie eine Hybrid-Hosting-Lösung	303
Konfigurieren Sie die Platzierung der Spielsitzungen	306
Eigenschaften der Warteschlange	307
Best Practices	308
Erstellen einer Warteschlange	309
Richten Sie die Ereignisbenachrichtigung ein	314
Passen Sie Ihre Game-Hosting-Lösung an	318
Spieleserver-Builds	319
Spielersitzungen und Matchmaking	333
Platzierungen von Spielsitzungen	334
Hosting-Ressourcen	352
Verwaltung einer Game-Hosting-Lösung	378
Ressourcen zum Hosten von Spielen verwalten	378
Amazon GameLift ServersFunktionen der Konsole	378
Konsolen-Dashboard	379
Baut	380
Flotten	384
Aliase	407
Spielsitzungs-Warteschlange	411
Bereite dich auf den Start des Spiels vor	414
Vorbereitung	415
Bereite dich auf den Test vor	415
Bereite dich auf den Start vor	416

Planen Sie für die Zeit nach der Markteinführung	416
Verwalte Game-Hosting-Ressourcen mit CloudFormation	417
Best Practices	417
Verwendung von CloudFormation Stacks	418
Builds werden aktualisiert	423
DDoS-Schutz	426
Wichtigste Vorteile	426
So funktioniert das Player-Gateway	427
Datenverkehrsfluss	427
Schlüsselkonzepte	427
Überwachen des Spieler-Gateways	429
IPv4 und IPv6 Kompatibilität	429
Aktiviere das Spieler-Gateway auf Flotten	429
Voraussetzungen	430
Player-Gateway-Modi	430
IPv4 und IPv6 Kompatibilität	430
Integrieren Sie das Player-Gateway in ein Spiel	431
Backend-Integration	431
Kundenintegration	433
Testen Sie Ihre Integration	435
Best Practices	435
Testen und Problembehebung	437
Richten Sie sich für iterative Entwicklung ein	437
Erstellen Sie eine cloudbasierte Testumgebung	439
Richten Sie lokale Tests ein	443
Richten Sie lokale Tests ein (Legacy)	449
Debuggen von Flottenproblemen	458
Probleme bei der Erstellung von Flotten	458
Probleme mit dem Serverprozess	460
Probleme bei der Flottenlöschung	460
Stellen Sie eine Remoteverbindung zu Flotteninstanzen her	461
Fernzugriff über die Konsole	462
Fernzugriff mit der AWS CLI	463
Connect zu einer Instanz herstellen (Server-SDK 5)	465
Stellen Sie eine Connect zu einer Instanz her (Server-SDK 4.x oder früher)	467
Dateien auf Remote-Instances anzeigen	470

Skalierung	472
Um die Flottenkapazität in der Konsole zu verwalten	473
Legen Sie Kapazitätsgrenzen für das Hosting fest	473
So legen Sie die Grenzwerte für die Flottenkapazität fest	474
Manuelles Festlegen der Flottenkapazität	475
Aussetzen des Auto Scaling	475
So legen Sie die Flottenkapazität manuell fest	476
Automatische Skalierung der Flottenkapazität	477
Zielbasiertes Auto Scaling	478
Regelbasierte auto Skalierung	480
Managen Sie die Skalierung von Fleet To/From Zero	485
Verhalten bei der Skalierung	486
Verhalten bei horizontaler Skalierung	486
Scale Zero konfigurieren To/From	486
Best Practices	488
Containerflotten skalieren	488
Überwachen	491
Vergleich der Metriken	492
Verfügbarkeit von Metriken nach Quelle	492
Topics	496
Überwachen Sie mit CloudWatch	496
Metriken, Dimensionen	497
Flottenmetriken	498
DDoMetriken zum S-Schutz (Spieler-Gateway)	515
Warteschlangenmetriken	518
FlexMatch-Metriken	522
FleetIQ-Metriken	526
Überwachung mit Server-Telemetrie-Metriemetriemetri	528
Vorteile von Telemetrie-Metriken	529
Bevor Sie beginnen	530
Implementierung	531
Verfügbare Metriken	532
Funktionsweise	537
Organisation und Nutzung des Dashboards	538
Allgemeine Überwachungsszenarien	550
Anleitung zur Fehlerbehebung	555

Protokollieren von API-Aufrufen	558
Amazon GameLift Servers-Informationen in CloudTrail	558
Grundlagen zu Amazon GameLift Servers-Protokolldateieinträgen	559
Servernachrichten protokollieren	561
EC2 im Vergleich zur Container-Protokollierung	562
Protokollierung für benutzerdefinierte Server	564
Sicherheit	568
Datenschutz	569
Verschlüsselung im Ruhezustand	571
Verschlüsselung während der Übertragung	571
Richtlinie für den Datenverkehr zwischen Netzwerken	572
Identity and Access Management	573
Zielgruppe	573
Authentifizierung mit Identitäten	574
Verwalten des Zugriffs mit Richtlinien	575
Funktionsweise von Amazon GameLift Servers mit IAM	577
Beispiele für identitätsbasierte Richtlinien	584
Fehlerbehebung	590
AWS verwaltete Richtlinien	592
Protokollierung und Überwachung mit Amazon GameLift Servers	594
Compliance-Validierung	595
Ausfallsicherheit	595
Sicherheit der Infrastruktur	596
Konfigurations- und Schwachstellenanalyse	597
Bewährte Methoden für die Gewährleistung der Sicherheit	598
Halten Sie die Runtime-Umgebungen Ihrer Flotte auf dem neuesten Stand	598
Schützen Sie Ihre Portkonfigurationen	600
Zusätzliche Sicherheitsressourcen	601
Preisgestaltung	602
Preismodelle	602
Generieren Sie Preisschätzungen	603
Schätzen Sie verwaltetes Hosting	603
Strategien zur Kostenoptimierung	607
Best Practices	607
Tools für Ressourcenkosten und -nutzung	609
Leistungsüberwachung	610

Referenzhandbücher	612
Service-API (AWS SDK)	612
Amazon GameLift ServersHosting-Ressourcen verwalten	613
Starte Spielsitzungen und trete Spielern bei	617
Server-SDK 5.x	618
Aktualisierungen des Server-SDK 5.x	619
Migrieren Sie zum Server-SDK 5.x	621
C++-Server-SDK	623
C#-Server-SDK	677
Gehe zum Server-SDK	725
C++ (Unreal) -Server-SDK: Aktionen	756
Server-SDK 4 und früher	805
Server-SDK SDK for C++: Aktionen	806
Server-SDK für C#: Aktionen	830
Server-SDK für Unreal: Aktionen	853
Veranstaltungen zur Platzierung von Spielsitzungen	869
Syntax für Platzierungsereignis	869
PlacementFulfilled	870
PlacementCancelled	872
PlacementTimedOut	873
PlacementFailed	874
AMI-Versionen	875
Servicestandorte	877
Unterstützte AWS Standorte	877
Standorte für verwaltetes Hosting	883
Standorte für überall Amazon GameLift Servers	884
Standorte für Amazon GameLift Servers FlexMatch	885
Amazon GameLift Serversin China	885
Service-Endpunkte und -kontingente	885
API-Limits für Amazon GameLift Servers	886
Referenz zu API-Grenzwerten	886
Drosselung auf Ressourcenebene	895
API-Beschränkungen für Amazon GameLift Servers SDK	895
UDP-Ping-Beacons	900
Wie funktionieren UDP-Ping-Beacons	900
Häufige Anwendungsfälle für UDP-Ping-Beacons	900

Beacon-Endpunkte abrufen	901
Implementierung von Latenzmessungen	902
Codebeispiele	903
Versionshinweise und SDK-Versionen	911
SDK-Versionen	911
Versionshinweise	921
.....	cmlxxiv

Was ist Amazon GameLift Servers?

Wird verwendet, Amazon GameLift Servers um dedizierte, kostengünstige Server in der Cloud für sitzungsbasierte Multiplayer-Spiele bereitzustellen, zu betreiben und zu skalieren. Die Lösung basiert auf einer AWS globalen Computerinfrastruktur und Amazon GameLift Servers unterstützt die Bereitstellung leistungsstarker, hochzuverlässiger Spieleserver bei gleichzeitiger dynamischer Skalierung Ihrer Ressourcennutzung, um der weltweiten Nachfrage der Spieler gerecht zu werden.

Schlüssel-Features

Globale Reichweite und geringe Latenz

- Bereite Hosting in AWS Regionen und Local Zones auf der ganzen Welt vor, um deine Spiele neuen Spielern zugänglich zu machen. Hier findest du eine vollständige Liste der [Standorte, die unterstützt werden von Amazon GameLift Servers](#).
- Sorgen Sie für niedrigstmögliche Latenz und ein hervorragendes Spielerlebnis, egal wo sich Ihre Spieler befinden.
- Verwenden Sie Platzierungswarteschlangen an mehreren Standorten, um Spieler dynamisch an Spielsitzungen mit optimalem Spielerlebnis teilhaben zu lassen.

Automatische Skalierung für verwaltete Flotten

- Optimieren Sie die Flottenkapazität im Hinblick auf Kosteneffizienz bei gleichbleibender Servicequalität.
- Skalieren Sie die Kapazität anhand der Spielernachfrage mithilfe von zielbasierter auto Skalierung
- Halten Sie den Kapazitätspuffer bereit, um plötzliche Spielerzuflüsse bewältigen zu können.

Flexible Hosting-Optionen

Weitere Informationen finden Sie unter [Amazon GameLift Servers Hosting-Optionen für Spiele](#).

- Spiele-Hosting in der Cloud, verwaltet von Amazon GameLift Servers
 - Hosten Sie Spieleserver auf der Infrastruktur von Amazon Elastic Compute Cloud (Amazon EC2). Wählen Sie zwischen Ressourcentypen wie rechenoptimierten (C-Familie), speicheroptimierten (R-Familie) und kostengünstigen ARM-basierten Graviton-Instances.

- Passen Sie an, wie alle Aspekte Ihrer Game-Hosting-Lösung Amazon GameLift Servers verwaltet werden, von der Bereitstellung von Spieleservern und dem Starten von Spielsitzungen bis hin zum Zuordnen von Spielern und deren Einbindung in Spiele.
- Verwenden Sie Flotten mit mehreren Standorten, um die globale Bereitstellung von Spieleservern zu optimieren.
- Nutzen Sie das kostengünstige Hosting mit [Amazon EC2 Spot-Instances](#).
- Stellen Sie Windows- oder Linux-Spieleserver mit verwaltetem EC2-Hosting bereit.
- Stellen Sie Linux-basierte Container auf Ihrem Spieleserver mit verwaltetem Container-Hosting bereit.
- Von Ihnen verwaltetes Game-Hosting mit Anywhere Amazon GameLift Servers
 - Nutzen Sie die Funktionen zur Platzierung von Amazon GameLift Servers Spielsitzungen und zur Sitzungsverwaltung.
 - Verwalte die Bereitstellung und Skalierung von Spieleservern auf deiner eigenen Hardware, deiner lokalen Infrastruktur oder anderen Cloud-Anbietern selbst.
- Hybrides Spiele-Hosting
 - Erstellen Sie eine Game-Hosting-Lösung, die eine Kombination aus Amazon GameLift Servers verwalteten Flotten und Anywhere-Flotten verwendet, um mit vorhandenen Lösungen zu arbeiten oder die Migration zu unterstützen.

Verwaltung von Spielsitzungen

- Lass uns die Prozesse auf den Spieleservern für dich Amazon GameLift Servers verwalten, den Status und die Verfügbarkeit von Spieleservern verfolgen und auf Anfrage neue Spielsitzungen platzieren.
- Nutze die Funktionen zur Platzierung von Spielsitzungen, um Spielern dabei zu helfen, Spielsitzungen zu finden und daran teilzunehmen, die das bestmögliche Spielerlebnis bieten. Verlassen Sie sich auf die Amazon GameLift Servers Entscheidungsfindung oder passen Sie die Einstellungen anhand von Kriterien wie Hosting-Kosten, Spielerlatenz und geografischem Standort an.
- Erstellen Sie Warteschlangen für die Platzierung von Spielsitzungen, um neue Anfragen für Spielsitzungen effizient zu bearbeiten. Verwenden Sie Funktionen, um ein Gleichgewicht zwischen Platzierungsqualität und Geschwindigkeit herzustellen, das Ihrem Spiel und den Erwartungen der Spieler entspricht.

- Nutzen Sie die optionalen Funktionen für Spielsitzungen, um Spielerplätze in Spielsitzungen zu reservieren, Spieler bei der Verbindung zu validieren und die Verfügbarkeit von Spielautomaten zu überwachen. Durchsuche oder filtere bestehende Spielsitzungen, um verfügbare Slots zu finden.

Erweitertes Matchmaking mit FlexMatch

- Fügen Sie robustes Matchmaking hinzu, das vollständig in Ihre Amazon GameLift Servers Hosting-Lösung integriert ist.
- Erstelle individuelle Regeln für die Spielsuche auf der Grundlage von Spielerattributen und -präferenzen, wie dem Fähigkeitsniveau oder der Charakterauswahl, und verwende bei Bedarf komplexe Logik.
- Richten Sie teambasierte Spiele für kompetitives oder kooperatives Gameplay ein. Gleichen Sie die and/or Spielerzusammensetzung der einzelnen Teamspiele aus.
- Optimiere den Matchmaking-Algorithmus, um die Qualität der Spiele und die Wartezeiten in Einklang zu bringen.
- Support Spiele von 2 bis 200 gleichzeitigen Spielern.
- Support Match-Backfilling, um verfügbare Spielerplätze zu füllen und die Spielsitzungen voll zu halten.

Umfassende Überwachung

- Erhalten Sie Echtzeit-Metriken zur Nutzung und Leistung von Spielsitzungen und Spieleraktivitäten sowie zur Hosting-Leistung und zum Zustand verwalteter Flotten.
- Integrieren Sie Amazon CloudWatch für Benachrichtigungen und Dashboards.
- Greifen Sie zur Fehlerbehebung auf Serverprotokolle und Event-Tracking zu.
- Analysieren Sie Leistungs- und Nutzungsdaten, um die Skalierung (verwaltete Flotten) oder die Ressourcenzuweisung (selbstverwaltete Flotten) zu optimieren.

Anpassung und Erweiterbarkeit

- Passen Sie die Logik der Platzierung von Spielsitzungen an, einschließlich der Priorisierung von Latenz, Kosten und Standort.

- Wählen Sie für verwaltetes Hosting Ihre Spielhosting-Ressourcen aus und geben Sie Anweisungen zum Starten und Ausführen von Spieleservern auf jeder Instanz. Passen Sie bei verwalteten Containern die Ressourcenzuweisung zwischen Containern genau an.
- Definieren Sie Ihre eigenen Matchmaking-Regeln.
- Implementieren Sie benutzerdefinierte Strategien für die Kapazitätsskalierung.
- Fügen Sie Spieleservercode hinzu, um ihn in Ihre vorhandenen AWS Dienste wie Datenbanken oder Tools zur Speicherung von Inhalten zu integrieren.
- Erstellen Sie einen Backend-Dienst mit benutzerdefinierten Funktionen, um Anfragen zum Beitritt von Spielern (z. B. Spielerpartys) zu verwalten, die Authentifizierung zu verwalten oder benutzerdefinierte Logik und Persistenz hinzuzufügen.

Integration mit dem Ökosystem AWS

- Verwenden Sie Dienste wie Amazon DynamoDB, Amazon Simple Storage Service und Amazon Aurora DSQL für die Persistenz des Spielstatus.
- Implementieren Sie Amazon Cognito für die Spielerauthentifizierung.
- Verarbeiten Sie Spielanalysen mit Amazon Kinesis und Amazon S3.
- Fügen Sie Voice-Chat mit Amazon Chime SDK hinzu.
- Erstellen Sie benutzerdefinierte Spielfunktionen mithilfe eines AWS Lambda Amazon API Gateway.

Vorgehensweise bei der Arbeit mit Amazon GameLift Servers

Verwenden Sie diese Tools für die Arbeit mit Amazon GameLift Servers.

AWS CLI

Verwenden Sie AWS Command Line Interface (AWS CLI), um Aufrufe an das AWS SDK zu tätigen, einschließlich der Service-API für Amazon GameLift Servers. Weitere Informationen finden Sie unter [Erste Schritte mit dem AWS CLI](#) im AWS Command Line Interface Benutzerhandbuch.

Amazon GameLift Servers-Konsole

Verwenden Sie das [AWS-Managementkonsole Formular](#), Amazon GameLift Servers um Ressourcen zu konfigurieren, Ihre Spieleserver-Bereitstellungen zu verwalten und Leistungs- und Nutzungsmetriken zu verfolgen. Die Amazon GameLift Servers Konsole ist eine GUI-Alternative zur programmgesteuerten Verwaltung von Ressourcen oder mit dem. AWS CLI

Amazon GameLift Servers SDKs

Sie Amazon GameLift Servers SDKs enthalten die Bibliotheken, die für die Kommunikation zwischen Ihren Spielclients, Spieleservern und Spieldiensten und dem Amazon GameLift Servers Dienst erforderlich sind. Weitere Informationen finden Sie unter [Holen Sie sich Amazon GameLift Servers Entwicklungstools](#).

AWS CloudFormation

Wird verwendet, AWS CloudFormation um AWS Ressourcen für Ihre Game-Hosting-Lösung zu modellieren, bereitzustellen und zu verwalten, indem Infrastruktur als Code behandelt wird. Erstellen Sie Vorlagen, die die Ressourcen beschreiben, und CloudFormation automatisieren Sie die Aufgaben der Konfiguration und Bereitstellung von Ressourcen an den von Ihnen angegebenen Speicherorten.

Amazon GameLift ServersHosting-Optionen für Spiele

Amazon GameLift Servers bietet eine Reihe von Optionen für das Hosten von Spieleservern für sitzungsbasierte Multiplayer-Spiele. Ganz gleich, ob Sie Cloud-basiertes Hosting einrichten möchten, das von Ihnen verwaltet wird, Amazon GameLift Servers oder Hosting-Ressourcen integrieren möchten, die Sie bereits haben, können Sie damit zusammenarbeiten, Amazon GameLift Servers um die Hosting-Lösung zu entwickeln, die Sie für Ihre Spieler benötigen. Weitere Informationen darüber, wie eine Game-Hosting-Lösung aussieht, finden Sie unter [So Amazon GameLift Servers funktioniert das Hosten mit](#).

Themen

- [Verwaltetes EC2](#)
- [Verwaltete Container](#)
- [Überall, Hosting](#)
- [Hybrides Hosting](#)

Verwaltetes EC2

Verlagern Sie die Arbeit für die Verwaltung Ihrer Spieleserver in der Produktionsumgebung auf Amazon GameLift Servers verwaltetes EC2-Hosting. Nutzen Sie die Optimierungen für Multiplayer-Spieleserver und verlassen Sie sich gleichzeitig auf die hohe Leistung und Zuverlässigkeit von Amazon Elastic Compute Cloud (Amazon EC2) und die AWS globale Computerinfrastruktur.

[Beginnen Sie mit dem Aufbau einer Amazon GameLift Servers verwalteten EC2-Hosting-Lösung](#)

Merkmale

- Hosten Sie Multiplayer-Spieleserver, die auf Amazon Linux- oder Windows Server-Betriebssystemen laufen.
- Wird verwendet Amazon GameLift Servers, um Ihre benutzerdefinierten Spieleserver auf Hosting-Ressourcen bereitzustellen und zu verwalten AWS Cloud. Wählen Sie den Hardwaretyp und den Einsatzort aus und konfigurieren Sie zusätzliche Details nur dort, wo Sie sie benötigen.
- Konfigurieren Sie den Runtime Process Manager so, dass die Spielserversprozesse automatisch verwaltet werden, wenn sie für das Hosten von Spielsitzungen erforderlich sind.
- Richten Sie ein benutzerdefiniertes Health Tracking ein, um Amazon GameLift Servers Spieleserver mit schlechter Leistung zu erkennen und zu beheben.
- Nutzen Sie die Vorteile der Amazon GameLift Servers Leistungsüberwachung. Verfolgen Sie Messwerte wie die Hardwareleistung und die Lebenszyklen von Serverprozessen. Sie können auch Sitzungsprotokolle von Spielen herunterladen und speichern.
- Wird verwendet Amazon GameLift Servers, um den Zugriff zwischen Spielserversprozessen und anderen AWS Ressourcen zu verwalten.
- Automatisieren Sie beim Hosten in der Produktion die Bereitstellung und Verwaltung von Spiele-Hostings mithilfe von AWS CloudFormation Vorlagen und dem AWS Cloud Development Kit (AWS CDK). Nutzen Sie Tools und Dienste für kontinuierliche Integration und kontinuierliche Bereitstellung (CI/CD) wie AWS CodePipeline

Verwaltete Container

Amazon GameLift Servers bietet eine komplette Cloud-Hosting-Lösung für containerisierte Spieleserver, sodass Sie die wichtigsten Vorteile der Container-Nutzung wie Portabilität, Agilität und Fehlertoleranz nutzen können. Wie beim verwalteten EC2-Hosting werden beim verwalteten Container-Hosting Ihre Container auf Amazon EC2 EC2-Instances bereitgestellt und ausgeführt.

[Beginnen Sie mit der Entwicklung einer Amazon GameLift Servers verwalteten Hosting-Lösung für Ihren containerisierten Spieleserver.](#)

Merkmale

- Entwickeln Sie eine benutzerdefinierte Architektur mit schlanken Containern, um den Build Ihres Spieleservers, die Abhängigkeiten und die Zusatzsoftware auszuführen.

- Verwenden Sie die Docker-Tools, um ein Linux-basiertes Container-Image zu erstellen. Speichern Sie Bilder zur Bereitstellung in einem Amazon Elastic Container Registry (Amazon ECR) - Repository.
- Wird verwendet Amazon GameLift Servers, um Ihre benutzerdefinierten Spieleserver auf Hosting-Ressourcen AWS Cloud bereitzustellen und zu verwalten. Wählen Sie den Hardwaretyp und den Einsatzort aus und konfigurieren Sie zusätzliche Details nur dort, wo Sie sie benötigen.
- Verwalte den Lebenszyklus deiner Hosting-Flotte mit Tools zur Modellierung von Spielserverversionen. Stellen Sie nach Bedarf Spielserver-Updates und andere Konfigurationsänderungen bereit.
- Verfolge die Leistung des Spiele-Hostings mit containerspezifischen Leistungskennzahlen. Überwachen Sie den Zustand Ihrer Flottenressourcen mithilfe von Hardwaremetriken.
- Verwalten Sie die Ressourcen Ihrer Containerflotte mithilfe von AWS CloudFormation Vorlagen für Amazon GameLift Servers.

Überall, Hosting

Mit Anywhere-Hosting können Sie Funktionen zur Verwaltung von Amazon GameLift Servers Spielsitzungen nutzen, einschließlich Matchmaking und Spielsitzungsmetriken, um Spieleserver zu hosten, wo immer Sie möchten. Verwalte deine Spieleserver-Bereitstellungen, die Überwachung des Zustands deiner Spieleserver und die Kapazitätsskalierung selbst.

Note

Anywhere-Flotten eignen sich besonders als Testumgebungen für eine schnelle, iterative Spieleentwicklung. Richten Sie eine Anywhere Flotte für Ihre eigene lokale Workstation oder eine Cloud-basierte Hosting-Ressource ein. Weitere Informationen zum Testen mit Anywhere finden Sie unter [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#).

[Beginnen Sie mit der Entwicklung einer Amazon GameLift Servers Anywhere-Hosting-Lösung für Ihr Spiel.](#)

Merkmale

- Stellen Sie Spieleserver auf Ihrer eigenen Hardware, Ihrer lokalen Infrastruktur oder anderen Cloud-Anbietern bereit. Nutze verfügbare Hardware in der Nähe deiner Spieler, überall.

- Verwenden Sie Funktionen zur Amazon GameLift Servers Sitzungsverwaltung, um die Verfügbarkeit von Spieleservern zu überwachen, neue Spielsitzungen zu starten und Spieler zu Spielsitzungen hinzuzufügen.

Hybrides Hosting

Für Produktionshosting können Sie sich für einen hybriden Ansatz entscheiden, bei dem Anywhere Flotten für die von Ihnen bereitgestellten und verwalteten Hosting-Ressourcen bereitgestellt und Amazon GameLift Servers verwaltete Flotten bei Bedarf eingesetzt werden. Eine Hybridlösung verwendet dieselben Prozesse, um Spielsitzungen zu starten und Spieler miteinander zu verbinden, um ein nahtloses Spielerlebnis zu bieten. Sie können Nutzungsmetriken für alle Hostings unabhängig vom Ressourcentyp überwachen und sammeln.

Häufige Szenarien, in denen eine Hybridlösung sinnvoll ist:

- Erweitern Sie Ihre Hosting-Lösung auf die AWS Cloud. Ergänzen Sie die Funktionen Ihrer vorhandenen Hosting-Lösung (lokale Hardware oder anderes cloudbasiertes Hosting) durch das Hinzufügen von Amazon GameLift Servers verwaltetem Hosting. Mit verwaltetem Hosting können Sie Ihre Hosting-Kapazität erhöhen oder „Burst“-Kapazität hinzufügen, um schnell zu skalieren und nur für Ressourcen zu zahlen, wenn Sie sie benötigen. Sie können auch die globale Präsenz des Amazon GameLift Servers Dienstes nutzen, um mehr Spieler auf der ganzen Welt zu erreichen und ihnen das Multiplayer-Erlebnis mit niedriger Latenz zu bieten, das sie erwarten.
- Bereiten Sie sich auf die Migration zum Cloud-basierten Hosting vor. Wenn Sie eine Migration auf das in Betracht ziehen oder planen AWS Cloud, ist eine Hybrid-Hosting-Lösung eine praktikable Möglichkeit, den Übergang so schrittweise wie nötig durchzuführen. Damit können Sie die Lösung entwickeln, die Sie gerade benötigen, und sich gleichzeitig darauf vorbereiten, wo Sie in future sein möchten.
- Geringere Latenz für Spieler an Standorten außerhalb der Standorte von Amazon GameLift Servers. Wenn du bereits Amazon GameLift Servers Managed Hosting verwendest, musst du möglicherweise Spieler in ungewöhnlichen Situationen unterstützen. Vielleicht möchtest du Spieler an ungewöhnlich abgelegenen Orten besser erreichen oder Spieler mit besonderen Zugangsbedürfnissen unterstützen. Fügen Sie benutzerdefinierte Hosting-Standorte hinzu und verwenden Sie Amazon GameLift Servers Anywhere, um diese Standorte zusammen mit Ihren verwalteten Hosting-Ressourcen zu unterstützen.
-

Beginne mit der Entwicklung einer Amazon GameLift Servers Hybrid-Hosting-Lösung für dein Spiel.

So Amazon GameLift Servers funktioniert das Hosten mit

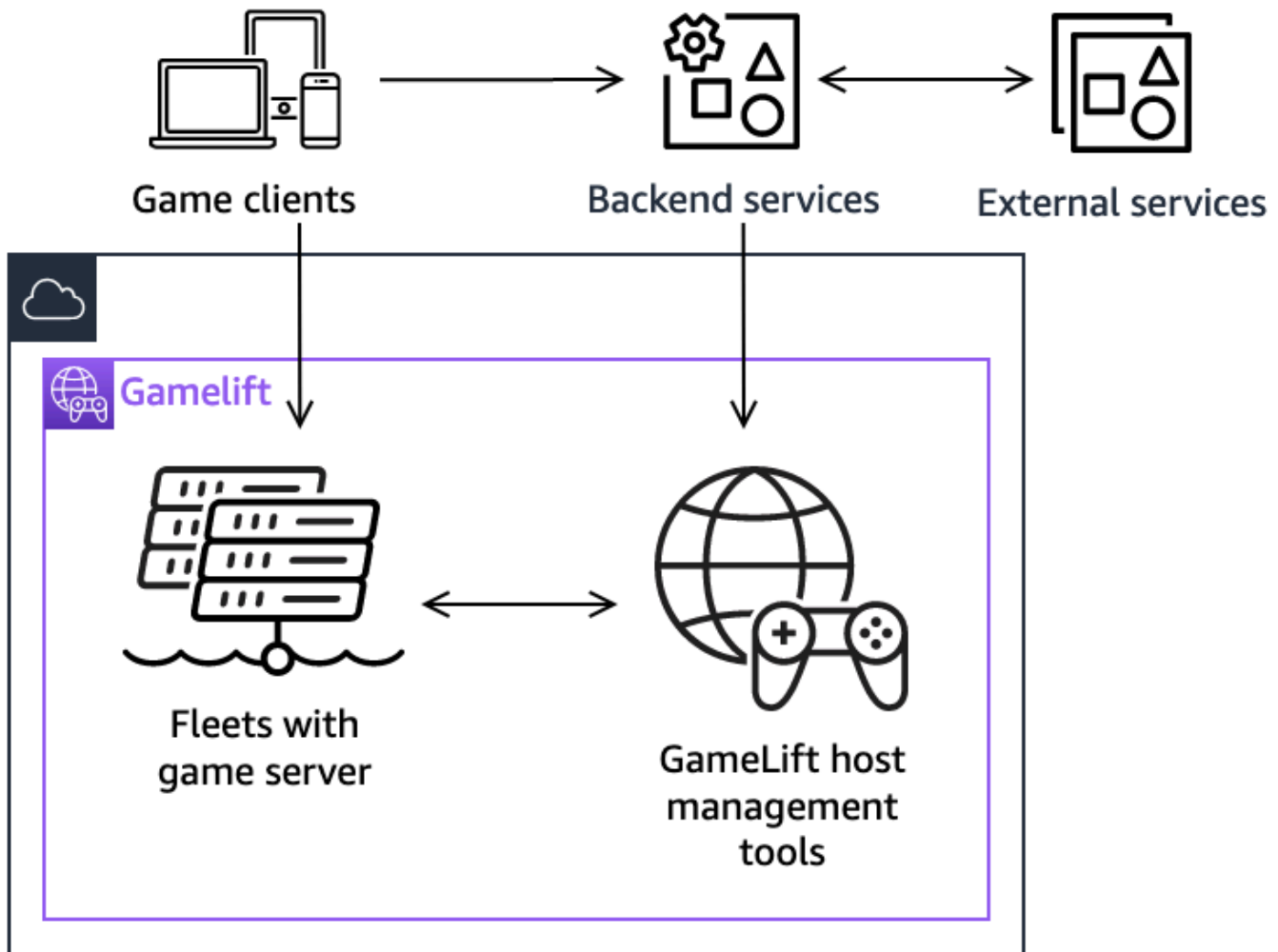
Amazon GameLift Servers ist ein Dienst, der dedizierte, kostengünstige Server, Infrastrukturbereitstellung, Skalierung und Sitzungsverwaltung für Ihre sitzungsbasierten Multiplayer-Spiele bereitstellt. Der Service bietet flexible Tools und Funktionen, die Sie an Ihre Spiele anpassen oder in Zusammenarbeit mit Ihren eigenen Systemen verwenden können. Nutze die Vorteile von Amazon GameLift Servers verwaltetem Hosting, damit du dich auf die Erstellung von Spielerlebnissen konzentrieren kannst, anstatt den Serverbetrieb zu verwalten.

In diesem Thema werden die Kernkomponenten einer vollständigen Amazon GameLift Servers Hosting-Lösung beschrieben und wie sie zusammenarbeiten, um Spielern Multiplayer-Spielsitzungen bereitzustellen.

Kernkomponenten einer Game-Hosting-Lösung

Eine komplette Amazon GameLift Servers Hosting-Lösung besteht aus mehreren wesentlichen Komponenten, die zusammenarbeiten, um Multiplayer-Spielerlebnisse zu bieten. Einige dieser Komponenten werden von Ihnen erstellt und verwaltet, während andere von Ihnen bereitgestellt werden und auf der Grundlage Ihrer Konfigurationsauswahl verwaltet werden. Für die Entwicklung einer effektiven Game-Hosting-Lösung ist es entscheidend zu verstehen, wie diese Komponenten interagieren und welche Rollen sie jeweils spielen. Zu den Komponenten gehören:

- Der Spielclient läuft auf Spielergeräten.
- Der Backend-Dienst ermöglicht es Spieleclients, mit ihnen zu kommunizieren Amazon GameLift Servers, um Informationen zur Spielsitzung abzurufen und an Spielen teilzunehmen.
- Die Gameserver-Software läuft auf Spielhosting-Ressourcen und hostet Spielsitzungen für eine Gruppe von Spielern.
- Das Platzierungssystem initiiert Spielsitzungen und ordnet Spieler den Spielen zu.
- Spiele-Hosting-Flotten bieten Spieleserver an einem oder mehreren geografischen Standorten.
- Das Game-Hosting-Managementsystem überwacht den Status des Spielhostings und verwaltet die Kapazität.



Spielclient

Der Spielclient ist deine Spielesoftware, die auf dem Gerät eines Spielers läuft. Er leitet die Spielerplatzierung in eine Spielsitzung ein, indem er mit einem Backend-Dienst kommuniziert und stellt eine direkte Verbindung zu einem Spieleserver her, um am Gameplay teilzunehmen.

Wichtige Funktionen für das Hosten von Spielen

- Senden Sie Beitrittsanfragen an den Backend-Dienst. Geben Sie relevante Spielerdaten und Spielsitzungsdaten als Teil der Beitrittsanfrage an.
- Erfassen Sie Latenzdaten für den Spielclient und nehmen Sie diese Informationen in die Beitrittsanfrage auf.
- Empfangen Sie Verbindungsinformationen zur Spielsitzung vom Backend-Dienst und verwenden Sie diese, um eine Verbindung zu einem Spieleserver herzustellen.

- Bewältigen Sie Verbindungsszenarien wie Spielerverifizierung, Annahme von Spielen Amazon GameLift Servers FlexMatch, Sitzungsunterbrechungen oder Verbindungsabbrüche.

Wer baut es

Sie fügen Ihrer Game-Client-Software Game-Hosting-Funktionen hinzu und richten die Kommunikation mit dem Backend-Dienst ein.

Backend-Dienst

Der Backend-Service ist eine Koordinationsschicht zwischen Spielclients und dem Amazon GameLift Servers Dienst. Er steuert die gesamte Kommunikation mit dem Dienst, um Anfragen nach Platzierungen für Spielsitzungen zu stellen und Informationen zu Spielsitzungen und Spielersitzungen abzurufen. Die Verwendung eines Backend-Dienstes ist eine bewährte Methode, um die sichere Kommunikation mit dem Dienst im Namen der Spieleclients aufrechtzuerhalten und zu vermeiden, dass sensible AWS Anmeldeinformationen und Berechtigungen an Spieleclients weitergegeben werden müssen.

Ihr Backend-Service implementiert eine Strategie zur Gruppierung von Spielern für Ihre Spielsitzungen. Sie haben viel Flexibilität, wenn es darum geht, wie Spieler in einer Spielsitzung landen. Sie können einen Browser für Spielsitzungen für Spieler einrichten oder von Spielern definierte Gruppen unterstützen. Oder Sie können Anfragen für Spielsitzungen bündeln, sobald sie eingehen, Spielergruppen bilden (möglicherweise mit Vorsortierung) und für jede Gruppe eine Spielsitzung starten. Mit der FlexMatch Spielerzuweisung kannst du Spielermatches bilden und bestehende Matches ergänzen.

Die wichtigsten Funktionen für das Hosten von Spielen

- Authentifizieren Sie die Kommunikation von Spieleclients.
- Stellen Sie Anfragen Amazon GameLift Servers zum Starten neuer Spielsitzungen, zum Abrufen von Informationen zu Spielsitzungen oder zum Beitritt zu bestehenden Spielsitzungen. Nehmen Sie bei Bedarf relevante Spielsitzungen und Spielerdaten in Ihre Anfragen auf.
- Erstellen Sie optional Anfragen für Spielersitzungen, um die Verfügbarkeit von Spielsitzungen genauer verfolgen zu können. Verwenden Sie Spielersitzungen, um Plätze für Spielsitzungen zu reservieren, Spieler zu überprüfen, wenn sie sich verbinden, und um Verbindungsabbrüche zu verfolgen.
- Antworten Sie den Spielclients auf Anfrage mit Verbindungsdetails für Spielsitzungen und anderen Informationen.

Wer baut es

Sie erstellen den Backend-Service in einer Umgebung, die von Ihnen verwaltet wird, um Ihre Spielanforderungen zu unterstützen und zu implementieren, wie Spieler in Spiele einsteigen. Integrieren Sie das AWS SDK, um Aufrufe an die Amazon GameLift Servers Service-API zu tätigen. Optional können Sie Ihren Backend-Service AWS mithilfe von Diensten wie AWS Lambda Amazon Simple Storage Service (Amazon S3) und Amazon Cognito aufbauen.

Spieleserver

Der Gameserver ist deine benutzerdefinierte Serversoftware, die den Spielstatus verwaltet, Spieleraktionen verarbeitet und das Gameplay mehrerer verbundener Spieler synchronisiert. Der Spieleserver unterhält die Kommunikation mit dem Amazon GameLift Servers Dienst, um das Hosting von Spielsitzungen zu verwalten.

Schlüsselfunktionen für das Hosten von Spielen

- Kommunizieren Amazon GameLift Servers Sie mit:
 - Status melden (bereit, Sessions zu veranstalten, bereit, Spieler aufzunehmen, Gesundheitszustand).
 - auf Serviceanrufe antworten (Spielsitzungen starten oder beenden).
- Verwalte den Lebenszyklus einer Spielsitzung, um pro Vorgang jeweils eine Spielsitzung zu veranstalten.
- Koordinieren Sie sich mit anderen AWS Diensten, um zusätzliche Funktionen zu erhalten.
- Bestätigen Sie optional die Verbindungen neuer Spieler.

Wer baut es

Du baust deine Gameserver-Software. Sie integrieren das Server-SDK Amazon GameLift Servers und fügen Funktionen hinzu, um eine Verbindung mit dem Dienst herzustellen und die Verwaltung von Spielsitzungen zu unterstützen.

Flotte, die Spiele hostet

Die Hosting-Flotte ist eine Sammlung von Computerressourcen, auf denen Ihre Spieleserver betrieben werden. Flottenressourcen können auf mehrere geografische Standorte verteilt werden, um Spielern unabhängig von ihrem Standort ein Gameplay mit geringer Latenz zu bieten. Jede

Flottenressource führt einen oder mehrere Spielserverprozesse aus, mit denen direkt kommuniziert wird. Amazon GameLift Servers Jeder Spielserver-Prozess kann jeweils eine Spielsitzung hosten.

Die Eigenschaften und Funktionen der Flotte variieren je nach Hosting-Typ der Flotte. Verwaltete Flotten stellen Ressourcen bereit AWS Cloud und werden von ihnen verwaltetAmazon GameLift Servers. Anywhere-Flotten sind vom Kunden bereitgestellte Rechenressourcen, die außerhalb von verwaltet werden. Amazon GameLift Servers

Die wichtigsten Funktionen für das Hosten von Spielen

- Stellen Sie Hosting-Ressourcen für Spieleserver bereit.
 - Verwaltete Flotten stellen cloudbasierte Amazon EC2 EC2-Instances mit einer Vielzahl von Konfigurationseinstellungen bereit. Eine Flottenkonfiguration bestimmt die Rechenleistung jeder Instance in der Flotte, den physischen Standort der Instances und andere Details. Verwaltete Flotten fügen als Reaktion auf die Kapazitätsskalierung EC2-Instances hinzu oder entfernen sie
 - Flottenbereitstellungen werden überall selbst verwaltet. Die Flotte kann aus physischer Hardware oder anderen Cloud-Ressourcen bestehen und nach Bedarf konfiguriert werden.
- Installieren Sie die Laufzeitumgebung und die Gameserver-Software.
 - Verwaltete Amazon EC2 EC2-Flotteninstanzen werden mit Ihrem Spieleserver-Build und einem Amazon Machine Image (AMI) mit einer kompatiblen Laufzeitumgebung bereitgestellt.
 - Verwaltete Container-Flotteninstanzen werden mit Ihrem Gameserver-Build und einem containeroptimierten AMI mit Docker-Tools und anderen Komponenten für die Zusammenarbeit mit Amazon ECS bereitgestellt.
 - Überall, wo Flotten mit der von Ihnen ausgewählten Gameserver-Software und dem Betriebssystem bereitgestellt werden.
- Verwalte den Lebenszyklus von Spielserverprozessen. Wenden Sie vorkonfigurierte Runtime-Anweisungen an, um Prozesse auf jeder Flottenressource zu starten und zu stoppen.

Wer baut es

Das hängt vom Hosting-Typ der Flotte ab:

- Verwaltete Amazon EC2 EC2-Flotten: Sie laden Ihren Spieleserver-Build hochAmazon GameLift Servers, auf dem er für die Bereitstellung auf Flotteninstanzen gespeichert wird. Amazon GameLift Servers stellt die Infrastruktur der Flotte bereit. AWS Cloud Sie konfigurieren die Flotte und weisen sie an, wie Spielserver auf jeder Flotteninstanz ausgeführt werden sollen.

- **Verwaltete Containerflotten:** Sie verpacken die Build- und Laufzeitanweisungen für Ihren Spieleserver in ein Container-Image, das Sie zur Bereitstellung in Amazon ECR speichern können. Amazon GameLift Servers stellt die Infrastruktur der Flotte bereit. AWS Cloud stellt eine Container-Architektur bereit und konfigurieren die Flotte so, dass sie Ihre Container hostet.
- **Überall verfügbare Flotten:** Sie stellen die gesamte Infrastruktur bereit und verwalten die Bereitstellung der Serversoftware. Sie erstellen Anywhere-Flotten, mit denen Sie Ihre aktiven Game-Hosting-Ressourcen verbinden können. Amazon GameLift Servers

System zur Platzierung von Spielsitzungen

Das Platzierungssystem für Spielsitzungen sucht nach verfügbaren Spieleservern, auf denen neue Spielsitzungen abgehalten werden können. Das System verwendet Echtzeitinformationen über die Verfügbarkeit von Spielservern, um optimale Platzierungsentscheidungen zu treffen.

In Amazon GameLift Servers der Warteschlange ist der Hauptmechanismus für die Platzierung von Spielsitzungen. Eine Warteschlange für Spielsitzungen verwendet Algorithmen, die Sie konfigurieren können, um Spielsitzungen so zu platzieren, dass das bestmögliche Ergebnis erzielt wird. Sie können Platzierungen anhand von Faktoren wie den niedrigsten Hosting-Kosten und der geringsten Spielerlatenz priorisieren und Sie können eine Warteschlange für die Suche an mehreren geografischen Standorten konfigurieren. Als Alternative zu Warteschlangen kannst du eine bestimmte Flotte als Gastgeber für deine Spielsitzungen festlegen.

Die wichtigsten Funktionen für das Hosten von Spielen

- Verarbeiten Sie Anfragen zur Platzierung von Spielsitzungen, die Sie vom Back-End-Dienst erhalten haben.
- Nehmen Sie Platzierungen auf der Grundlage von Echtzeitinformationen zur Verfügbarkeit von Hosting-Ressourcen vor.
- Verwende Daten zur Spielerlatenz und andere Daten, um Platzierungsoptionen zu priorisieren (nur Warteschlangen)
- Fordere die Spieleserverprozesse auf, neue Spielsitzungen zu starten.
- Aktualisieren Sie die Verbindungsinformationen für die Spielsitzung, sobald die Spielsitzung bereit ist, Spieler aufzunehmen.
- Richten Sie optional die FlexMatch Spielerzuweisung ein, um Spielermatches zu erstellen und die Platzierung von Spielsitzungen für Spiele anzufordern.

Wer baut es

Amazon GameLift Servers liefert das Vermittlungssystem. Sie konfigurieren das Platzierungsverhalten anhand der Art und Weise, wie Ihr Backend-Service Platzierungsanfragen stellt, und optional, indem Sie Warteschlangen für Spielsitzungen und FlexMatch Matchmaker einrichten. Verwenden Sie die Amazon GameLift Servers Konsole, das AWS SDK oder den, AWS CLI um Warteschlangen und Matchmaker für dein Spiel zu erstellen und zu konfigurieren.

Managementsystem für das Hosting von Spielen

Das Game-Hosting-Management-System ist das operative Rückgrat, das alle Aspekte Ihrer Game-Hosting-Lösung koordiniert und überwacht. Dieses System bietet die Intelligenz und Automatisierung, die Amazon GameLift Servers Platzierungs- und Skalierungsfunktionen ermöglichen.

Schlüsselfunktionen für das Hosten von Spielen

- Verfolge den Status und die Verfügbarkeit von Spielserverprozessen, Spielsitzungen und Spielersitzungen in allen Flotten in Echtzeit, um die Platzierung von Spielsitzungen und die automatische Kapazitätsskalierung zu unterstützen.
- Überwachen Sie den Zustand und die Leistung Ihrer Flotte.
- Erfassen und analysieren Sie Kennzahlen zu den Aktivitäten beim Hosten von Spielen.
- Stellen Sie Tools zur Kapazitätsskalierung bereit, einschließlich automatischer Skalierung auf der Grundlage der Spielernachfrage und der Flottenauslastung.
- Verwalte Updates für Gameserver-Software und Laufzeitanweisungen.

Wer baut es

- **Verwaltete Flotten:** Amazon GameLift Servers bietet Verfügbarkeitsverfolgung, Metriken zu Game-Hosting-Aktivitäten und Hardwareleistung sowie Tools zur Kapazitätsskalierung. Sie verwenden die verfügbaren AWS Tools, um Softwareupdates für Spieleserver zu verwalten, Laufzeitanweisungen zu ändern und Ihre Verwendung von Messwerten individuell anzupassen (z. B. die Einrichtung CloudWatch für die Überwachung). Sie konfigurieren eine benutzerdefinierte Skalierungsrichtlinie und ändern sie nach Bedarf.
- **Anywhere-Flotten:** Amazon GameLift Servers bietet Statistiken zur Verfügbarkeitsverfolgung und zum Hosten von Spielen. Du verwaltest Änderungen an der Flottenkonfiguration, einschließlich Updates der Gameserver-Software und Laufzeitanweisungen. Sie erstellen Systeme zur Überwachung der Flottenleistung und zur Verwaltung der Flottenkapazitätsskalierung.

Wie die Komponenten zusammenarbeiten

Wenn ein Game-Hosting-Computer bereitgestellt wird

- Gameserver-Software installiert: Die Rechenleistung wird mit einer Laufzeitumgebung und Ihrem Gameserver-Build installiert
- Spieleserver wird gestartet: Auf dem Computer wird mindestens eine Instanz der ausführbaren Datei des Gameservers mit optionalen Startparametern gestartet.
- Spieleserver stellt eine Verbindung her mit Amazon GameLift Servers: Im Rahmen seiner Startaktionen ruft der Spieleserverprozess das Server-SDK auf, um eine Verbindung mit dem Dienst zu initialisieren.
- Der Spieleserver meldet sich bereit, eine Spielsitzung zu veranstalten: Der Spieleserver-Prozess schließt die Startaktionen ab und ruft dann das Server-SDK auf, um die Bereitschaft zu melden. Es beginnt, den Gesundheitsstatus auf der Grundlage seiner Konfiguration zu melden.
- Amazon GameLift Servers verfolgt die Verfügbarkeit: Der Dienst zeichnet die Verfügbarkeit des Spieleserverprozesses für die Platzierung von Spielsitzungen auf. Außerdem aktualisiert er Messwerte für Flotten, Instanzen und Spielsitzungen, um Nutzung und Kapazität nachzuverfolgen.
- Der Spieleserver wartet auf die Zuweisung der Spielsitzung: Der Spieleserverprozess behält seinen inaktiven Status bei, während er auf eine Aufforderung zum Starten einer Spielsitzung wartet.

Wenn ein Spieler einem Spiel beitreten möchte

- Spieler initiiert ein Spiel: Ein Spieler startet seinen Spielclient, authentifiziert sich bei deinem Backend-Service und bittet darum, einem Spiel beizutreten.
- Der Backend-Dienst fordert eine Spielsitzung an: Ihr Backend-Dienst ruft Amazon GameLift Servers an, um eine passende Spielsitzung zu finden oder zu erstellen, basierend auf seiner Strategie zur Gruppierung von Spielern. Die Anfrage kann Spieler- oder Spieldaten zur Verwendung in der Spielsitzung enthalten.
- Amazon GameLift Servers platziert die Spielsitzung: Wenn eine neue Spielsitzung gestartet wird, identifiziert das Platzierungssystem einen optimalen Ort für die Sitzung und wählt einen verfügbaren Spieleserver-Prozess aus. Der Dienst fordert den ausgewählten Prozess auf, eine neue Spielsitzung zu starten, und leitet alle Spieler- oder Spieldaten weiter. Wenn FlexMatch Matchmaking verwendet wird, erstellt der Matchmaker zuerst ein Spiel und fordert dann die Platzierung für das Spiel an.

- Der Spielserver startet die Spielsitzung: Der Spielserver leitet Schritte ein, um eine Spielsitzung zu starten. Wenn der Vorgang abgeschlossen ist, meldet er Amazon GameLift Servers, dass er bereit ist, Spielerverbindungen anzunehmen.
- Verbindungsinformationen werden an den Spielclient übermittelt: Nachdem der Spielserver seinen Status aktualisiert hat, werden die Verbindungsinformationen zur Spielsitzung Amazon GameLift Servers bereitgestellt. Der Backend-Dienst empfängt diese Informationen und übermittelt sie an den Spielclient.
- Der Spieler stellt eine Verbindung zur Spielsitzung her: Der Spielclient verwendet die Verbindungsinformationen, um eine direkte Verbindung zum Spielserver herzustellen und das Spiel zu starten.
- Amazon GameLift Servers überwacht den Status der Spielsitzung: Der Spielserverprozess meldet den Systemstatus, optional den Verbindungsstatus der Spieler und den Status der Spielsitzung, um die Verfügbarkeit der laufenden Spielsitzungen nachzuverfolgen.
- Der Spielserver-Prozess wird heruntergefahren: Der Spielserver-Prozess beendet die Spielsitzung, meldet den Status und fährt sich dann selbst herunter.

Amazon GameLift Servers und das Spielerlebnis

Spieler erwarten schnelle, faire und zuverlässige Multiplayer-Spielerlebnisse. Amazon GameLift Servers bietet diese Erlebnisse, indem Spieleentwicklern Tools zur Verfügung gestellt werden, die die Art und Weise, wie Spieler sich mit Multiplayer-Spielen verbinden und mit ihnen interagieren, direkt verbessern. In diesem Thema werden die wichtigsten Vorteile beschrieben, die Amazon GameLift Servers Funktionen für Spieler bieten.

Bieten Sie Spielern weltweit schnelles Gameplay

Spieler möchten mit minimaler Verzögerung eine Verbindung zu Servern in der Nähe herstellen. Amazon GameLift Servers sorgt durch intelligente Platzierung und globale Infrastruktur für optimale Verbindungserlebnisse:

- Spieler erleben weltweit eine gleichbleibende Leistung — Hosten Sie Spiele in AWS Regionen und Local Zones, die weltweit verfügbar sind, um die Entfernung zwischen Spielern und Spielservern zu minimieren (siehe [Amazon GameLift Servers Servicestandorte](#)). Sie können an diesen Standorten aus einer Vielzahl von EC2 Amazon-Instance-Typen wählen, um die besten Rechenressourcen zum Hosten Ihrer Spiele-Workloads zu finden.

- Spieler stellen eine Verbindung zu den besten verfügbaren Servern her. Dadurch können Spieler Amazon GameLift Servers an Spielsitzungen mit der geringstmöglichen Latenz teilnehmen, indem Netzwerkdaten des Spielclients gesammelt und an mehreren Standorten nach verfügbaren Servern gesucht werden. Verwenden Sie UDP-Ping-Beacons, um genaue Latenzdaten zu sammeln. Konfigurieren Sie Latenzrichtlinien, um die Verbindungsqualität mit akzeptablen Wartezeiten für Spieler in Einklang zu bringen.
- Spieler aus verschiedenen Regionen können zusammen spielen — Durch Platzierungen von Spielsitzungen können Spielsitzungen mit Spielern aus regionalen Spielerpools ausgeglichen und gleichzeitig eine akzeptable Latenz aufrechterhalten werden. Spieler aus weniger besiedelten Regionen können Spiele finden, ohne die Verbindungsqualität zu beeinträchtigen.

Hilf Spielern, schnell an Spielen teilzunehmen

Spieler möchten unabhängig von der Nachfrage schnellen Zugriff auf Spiele haben. Amazon GameLift Servers bietet eine Reihe von Tools zur Kapazitätsskalierung, mit denen Sie sicherstellen können, dass Spieler in Spitzenzeiten und in Ruhezeiten Spiele finden können:

- Spieler können verfügbare Server leicht finden — Mit Tools zur automatischen Skalierung kannst du sicherstellen, dass Hosting-Ressourcen bereit sind, wenn sie benötigt werden. Target Tracking verwaltet einen flexiblen Kapazitätspuffer, um die Nachfrage der Spieler nach deinem Spiel zu antizipieren und proaktiv zu skalieren.
- Spieler erleben eine reibungslose Leistung bei besonderen Ereignissen — Kombinieren Sie auto-scaling mit manuellen Skalierungstechniken, um plötzliche Zuflüsse nach geplanten Ereignissen wie Spielupdates, Marketingkampagnen oder Turnieren zu bewältigen.
- Spieler müssen nicht warten, bis die Server hochgefahren sind — Auf jeder Spiele-Hosting-Ressource sind die Spieleserver vorgewärmt, sodass sie immer bereit sind, neue Sessions zu veranstalten, sobald Spieler beitreten.

Entwickle kreative Methoden, um Spieler für Spielsitzungen zu gruppieren

Die Spieler möchten auf faire und ausgewogene Weise mit anderen konkurrieren oder mit ihnen zusammenarbeiten. Amazon GameLift Servers bietet eine Reihe von Optionen, um Spieler in Spielsitzungen zu gruppieren:

- Spieler werden gruppiert, um das Spielerlebnis zu optimieren. Mithilfe einer Reihe von Funktionen können Sie die Gruppierung von Spielern so verwalten, dass sie am besten zu Ihrem Spiel passen.

Erlaube es Spielern, Spielsitzungen zu suchen oder zu durchsuchen oder Spieleranfragen zu gruppieren, wenn sie ankommen, basierend auf Spielstärke, Latenz und anderen Attributen. Du kannst die Gruppierung von Spielern selbst verwalten oder die Spielerzuweisung mit anpassbaren FlexMatch Spielregeln implementieren.

- Spieler können aktiven Spielen beitreten — Bilden Sie Spielergruppen, um neue Spielsitzungen zu starten, oder nutzen Sie Spielersitzungen, um verfügbare Slots in bestehenden Sitzungen zu verfolgen und neue Spieler hinzuzufügen. Mit werden Sessions mit SpielernFlexMatch, die gut zueinander passen, automatisch aufgefüllt.
- Freunde können als Team zusammen spielen — Durch die Unterstützung von Spielergruppen können Gruppen von Freunden zusammen spielen. Spielsitzungen können die verbleibenden Plätze mit anderen Spielern füllen oder den Zugang einschränken. Mit FlexMatch können Freunde Teams bilden.
- Spieler können benutzerdefinierte Daten zu Spielsitzungen hinzufügen — Fügen Sie Spiel- und Spielerinformationen in Anfragen für Spielsitzungen und Spielersitzungen hinzu. Die Informationen werden zur Verwendung in einer Spielsitzung an den Spieleserver weitergegeben.
- Spieler warten nicht unendlich auf perfekte Spiele — Mit dieser Funktion können Sie die Spielanforderungen im Laufe der Zeit lockernFlexMatch, um zu verhindern, dass Spieler aufgrund übermäßiger Warteschlangen frustriert werden. Erstelle schrittweise Lockerungsregeln, die den wichtigsten Aspekten der Spielqualität Priorität einräumen.
- Spieler können wählen, ob sie Spiele annehmen oder ablehnen möchten — Mit können Sie Workflows so anpassenFlexMatch, dass Spieler potenzielle Spiele annehmen oder ablehnen können.

Sorgt während des gesamten Spiels für zuverlässige Verbindungen

Die Spieler erwarten ein konsistentes Gameplay ohne Unterbrechungen oder Verbindungsabbrüche. Amazon GameLift Serversbietet umfassende Sitzungsverwaltung und umfassenden Schutz:

- Spieler genießen hochwertige Leistung in der AWS Cloud — Mit dem von Amazon GameLift Servers verwalteten Hosting erhalten Spieler schnelle, zuverlässige Spieleserver, die auf einer AWS Computerinfrastruktur laufen.
- Spieler validieren, wenn sie sich verbinden — Wenn du Spielersitzungen für dein Spiel aktivierst, kannst du Spielerplätze in Spielsitzungen Amazon GameLift Servers reservieren und Spieler bei der Verbindung validieren.

- Spieler können die Verbindung wiederherstellen und den Spielfortschritt aufrechterhalten — Amazon GameLift Servers unterstützt die Wiederverbindung sowohl bei serverseitigen als auch bei clientseitigen Unterbrechungen.
- Die Spiele der Spieler sind vor Beendigung geschützt — Die optionale Funktion zum Schutz von Spielsitzungen verhindert, dass aktive Sitzungen bei Skalierungsereignissen oder anderen Unterbrechungen beendet werden.

Verbessern Sie das Spielerlebnis auf der Grundlage realer Daten

Die Spieler erwarten, dass Spiele konsistent funktionieren und mit der Zeit besser werden. Amazon GameLift Servers bietet umfassende Überwachungs- und Analysefunktionen, die zur Optimierung des Spielerlebnisses beitragen:

- Spieler erleben weniger Unterbrechungen — Durch die Echtzeitüberwachung der Flottenleistung, der Spielsitzungen und der Spieleraktivitäten können Probleme erkannt werden, bevor sie sich erheblich auf das Gameplay auswirken. Passen Sie das Herunterfahren und Migrieren von Spielsitzungen an.
- Spieler profitieren von einer proaktiven Überwachung des Serverzustands. Sie Amazon GameLift Servers ermöglicht eine kontinuierliche Überwachung des Serverzustands und ersetzt fehlerhafte Spieleserver automatisch, um die Auswirkungen von Hardware- oder Softwareausfällen zu minimieren. Konfigurieren Sie die Parameter für die Integritätsprüfung, um unterschiedliche Spielanforderungen zu unterstützen.
- Spieler profitieren von datengestützten Verbesserungen im Gameplay — Nutzen Sie die Sitzungsprotokolle und detaillierte Analysen, um Muster im Spielerverhalten und in der Serverleistung aufzudecken. Fügen Sie Unterstützung für benutzerdefinierte Protokollierung für spielspezifische Ereignisse hinzu.

Integrieren Sie erweiterte Spielfunktionen in andere Dienste AWS

Spieler wünschen sich integrierte Funktionen wie Sprachchat, sichere Authentifizierung und dauerhaften Spielfortschritt. Amazon GameLift Servers kann in andere AWS Dienste integriert werden, um ein umfassendes Spielerlebnis zu bieten:

- Spieler erhalten eine optimierte Authentifizierung — Die Amazon Cognito Cognito-Integration unterstützt verschiedene Identitätsanbieter und Authentifizierungsmethoden und

gewährleistet gleichzeitig sichere Spieleridentitäten über Spielsitzungen hinweg mit optimierten Anmeldeprozessen für wiederkehrende Spieler.

- Spieler können ihren Spielfortschritt über Sessions hinweg beibehalten — die Amazon DynamoDB DynamoDB-Integration kann verwendet werden, um Spielerfortschritte, Inventare und persistente Daten mit Hochleistungszugriff zu speichern, der minimale Auswirkungen auf das Gameplay gewährleistet und gleichzeitig die sitzungsübergreifende Kontinuität unterstützt.
- Spieler profitieren von analytischen Erkenntnissen — Die Integration von Amazon Kinesis und Amazon Simple Storage Service (Amazon S3) verarbeitet Spielanalysen, um Einblicke in das Spielerverhalten und die Vorlieben zu gewinnen. Dies ermöglicht Echtzeitanalysen, die Entwicklern helfen, sich an sich ändernde Spielmuster anzupassen und Spiele frisch und interessant zu halten.

Verwaltung des Game-Hostings in der Produktion mit Amazon GameLift Servers

Nachdem du dein Spiel mit gestartet hast Amazon GameLift Servers, musst du deine Game-Hosting-Infrastruktur verwalten, um eine optimale Leistung, Zuverlässigkeit und ein optimales Spielerlebnis zu gewährleisten. In diesem Thema werden die wichtigsten Funktionen und Tools für die Verwaltung der Lebenszyklen von Spiele-Hostings während der Produktion behandelt.

Überwachung des Zustands und der Leistung des Hostings

Amazon GameLift Servers bietet umfassende Überwachungstools, mit denen Sie den Zustand und die Leistung Ihrer Spiele-Hosting-Infrastruktur verfolgen können:

- Leistungskennzahlen — Überwachen Sie wichtige Leistungsindikatoren für verwaltete Flotten, wie z. B. die Nutzung von Instanzen, Spielersitzungen und Platzierungen von Spielsitzungen. Verfolgen Sie diese Kennzahlen in Amazon GameLift Servers oder in Amazon CloudWatch, wo Sie auch Alarmer einrichten können, um Benachrichtigungen zu erhalten, wenn Metriken Schwellenwerte überschreiten.
- Flottenkennzahlen und Ereignisse — Verfolgen Sie flottenspezifische Kennzahlen, einschließlich aktiver Serverprozesse, verfügbarer Spielsitzungen und aktiver Spielersitzungen, um eine optimale Kapazität sicherzustellen.
- Kennzahlen und Ereignisse zur Platzierung von Spielsitzungen — Verfolgen Sie Kennzahlen und Ereignisse, um den Zustand und die Leistung Ihres Systems zur Platzierung von Spielsitzungen zu überwachen.

- Spielsitzungsprotokolle — Greifen Sie auf Spielsitzungsprotokolle zu und analysieren Sie sie, um Probleme zu identifizieren und die Verhaltensmuster der Spieler zu verstehen.
- RealtimeSkriptprotokolle — Überwachen Sie bei Spielen, die Echtzeitserver verwenden, die Ausführung und Leistung von Skripten anhand detaillierter Protokolle.
- AWS Health Dashboard — Bleiben Sie über den AWS Servicestatus auf dem Laufenden, der sich auf Ihre Amazon GameLift Servers Bereitstellungen auswirken könnte.

Sie können über die Amazon GameLift Servers Konsole oder AWS CLI auf diese Überwachungstools zugreifen oder das AWS SDK verwenden, Amazon GameLift Servers um benutzerdefinierte Dashboards und Überwachungslösungen zu erstellen.

Verwaltung von Updates und Patches für Spieleserver

Ihre Spieleserver auf dem neuesten Stand zu halten, ist entscheidend für Sicherheit, Leistung und das Hinzufügen neuer Funktionen. Amazon GameLift Servers bietet verschiedene Methoden für die Verwaltung von Updates:

- Build-Verwaltung — Lade mehrere Versionen deiner Gameserver-Builds hoch und verwalte sie. Jeder Build ist versioniert und kann auf verschiedenen Flotten eingesetzt werden.
- Flottenaustausch — Erstelle neue Flotten mit AMI-Versionen und Spielserver-Build-Updates und verlagere mithilfe von Warteschlangen und Aliasnamen für Spielsitzungen den Traffic schrittweise von alten Flotten auf neue.
- Skript-Updates — Bei Echtzeitservern können Sie Serverskripte aktualisieren, ohne die gesamte Flotte durch das Hochladen neuer Skriptversionen ersetzen zu müssen.
- Container-Updates — Aktualisieren Sie bei containerbasierten Bereitstellungen Container-Images und Definitionen. Stellen Sie neue Versionen Ihrer Spieleserver für bestehende Flotten bereit. Ersetzen Sie Flotten, um AMI-Versionen zu aktualisieren.
- Automatisierte Bereitstellungen — Verwenden AWS CodePipeline und erstellen AWS CodeDeploy Sie CI/CD Pipelines für automatisierte Spielserver-Updates.

Erwägen Sie bei der Planung von Updates die Verwendung von blue/green Bereitstellungsstrategien, um Unterbrechungen für aktive Spieler so gering wie möglich zu halten und schnelle Rollbacks zu ermöglichen, falls Probleme festgestellt werden.

Optimierung von Leistung und Skalierung

Wenn sich Ihre Spielerbasis weiterentwickelt, müssen Sie Ihre Hosting-Konfiguration anpassen, um optimale Leistung und Kosteneffizienz zu gewährleisten:

- **Automatische Skalierung** — Konfiguriere Richtlinien zur Flottenskalierung auf der Grundlage von Kennzahlen wie Spielerzahl oder Auslastung der Spielsitzungen, um die Kapazität automatisch anzupassen.
- **Regionale Bereitstellungsanpassungen** — Fügen Sie je nach Spielerdemografie und Latenzanforderungen Regionen zu Ihren standortübergreifenden Bereitstellungen hinzu oder entfernen Sie sie aus ihnen.
- **Warteschlangenmanagement** — Optimiere die Konfiguration der Warteschlangen für die Platzierung von Spielsitzungen, um ein ausgewogenes Verhältnis zwischen Spielererlebnis und Hosting-Kosten zu erreichen.
- **Auswahl des Instanztyps** — Analysieren Sie Leistungskennzahlen, um die kostengünstigsten Instanztypen für Ihre Spieleserver-Anforderungen zu ermitteln.
- **Nutzung von Spot-Instances** — Nutzen Sie Spot-Instances für unkritische Workloads, um Kosten zu senken, und nutzen Sie geeignete Fallback-Strategien für On-Demand-Instances.
- **FlexMatchFeinabstimmung** — Verfeinern Sie die Regeln und Algorithmen für die Spielerzuweisung auf der Grundlage von aktuellen Spielerdaten und Feedback.

Überprüfe regelmäßig CloudWatch Kennzahlen und Kostenberichte, um Optimierungsmöglichkeiten zu identifizieren und Änderungen über die Amazon GameLift Servers Konsole oder API vorzunehmen.

Problembehebung und Live-Betrieb

Effektive Problembehebung und Live-Betriebsmanagement sind für die Aufrechterhaltung eines positiven Spielerlebnisses unerlässlich:

- **Benachrichtigungen über Flottenereignisse** — Richten Sie Amazon Simple Notification Service-Benachrichtigungen für Flottenereignisse wie Skalierungsaktivitäten, Instanzbeendigungen oder Fehler bei der Platzierung von Spielsitzungen ein.
- **Debugging der Platzierung von Spielsitzungen** — Verwenden Sie detaillierte Platzierungsprotokolle, um Probleme mit der Platzierung von Spielsitzungen zu identifizieren und zu lösen.

- **Integritätsprüfungen von Serverprozessen** — Überwachen Sie den Zustand von Serverprozessen und ersetzen Sie automatisch Prozesse, die die Zustandsprüfungen nicht bestehen.
- **Fernzugriff** — Connect zu Flotteninstanzen her, um sie direkt zur Fehlerbehebung zu verwenden AWS Systems Manager Session Manager.
- **Aliasverwaltung** — Verwenden Sie Aliase, um Spieler-Traffic schnell von problematischen Flotten wegzuleiten, ohne die Client-Konfigurationen zu ändern.
- **Backup und Wiederherstellung** — Implementieren Sie regelmäßige Backups kritischer Spieldaten und -konfigurationen, um eine schnelle Wiederherstellung nach Ausfällen zu ermöglichen.

Richten Sie klare Verfahren zur Reaktion auf Vorfälle und Runbooks für häufig auftretende Probleme ein, um Ausfallzeiten und Auswirkungen auf die Spieler zu minimieren.

Tools und Integration

Amazon GameLift Servers lässt sich in verschiedene AWS Dienste und Tools von Drittanbietern integrieren, um Ihre Produktionsmanagementfunktionen zu verbessern:

- **AWS CloudFormation**- Definieren und verwalten Sie Ihre GameLift Ressourcen als Infrastruktur als Code für konsistente Bereitstellungen.
- **AWS Lambda**- Erstellen Sie serverlose Funktionen, um routinemäßige Verwaltungsaufgaben zu automatisieren und auf Ereignisse zu reagieren.
- **Amazon EventBridge** — Erstellen Sie ereignisgesteuerte Architekturen, die automatisch auf Änderungen in Ihrer Umgebung reagieren. Amazon GameLift Servers
- **AWS SDK-Integration** — Verwenden Sie diese Option AWS SDKs , um benutzerdefinierte Verwaltungstools und Dashboards zu erstellen, die auf Ihre spezifischen Bedürfnisse zugeschnitten sind.
- **Überwachung durch Dritte** — Integrieren Sie mithilfe des CloudWatch Metrik-Exports in Überwachungs- und Analyseplattformen von Drittanbietern.

Nutzen Sie diese Integrationen, um eine umfassende Verwaltungslösung zu erstellen, die auf die Arbeitsabläufe und Anforderungen Ihres Teams abgestimmt ist.

Zugehörige Ressourcen

- [Amazon GameLift mit Amazon überwachen CloudWatch](#)

- [Aktualisierung Ihrer GameLift Amazon-Flotten](#)
- [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift](#)
- [Amazon GameLift Server SDK-Referenz](#)

Amazon GameLift Serverswichtige Begriffe

Dieses Thema enthält Definitionen für wichtige technische Begriffe, die in Amazon GameLift Servers Dokumentation und Diensten verwendet werden.

Aktive Spielsitzung

Eine Spielsitzung, die gerade läuft und Spieler akzeptiert oder hostet. Wird als Messgröße zur Überwachung der Flottenauslastung und Kapazitätsplanung verwendet. Siehe [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#).

Alias

Eine benutzerfreundliche Kennung für eine Flotte, die eine Abstraktionsebene zwischen deinem Spielclient und den Flottenressourcen bietet. Aliase ermöglichen es dir, Flottenkonfigurationen zu aktualisieren, ohne deinen Spielclient-Code zu ändern. Siehe [Zusammenfassung einer Amazon GameLift Servers Flottenbezeichnung mit einem Alias](#).

AMI (Amazon Machine Image)

Ein vorkonfiguriertes VM-Image, das zur Erstellung von EC2 Amazon-Instances für verwaltete Flotten verwendet wird. Enthält das Betriebssystem, die Laufzeitbibliotheken und die Amazon GameLift Servers Agentensoftware, die für den Betrieb von Spieleservern erforderlich sind. Siehe [Amazon GameLift Serversverwaltete EC2 Flotten](#).

Überall, Flotte

Eine Art von Flotte, die vom Kunden bereitgestellte Rechenressourcen verwendet, die außerhalb von Amazon GameLift Servers verwaltet werden. Mit Anywhere-Flotten können Sie bestehende Infrastruktur- oder Hybrid-Cloud-Lösungen in die Verwaltung von Amazon GameLift Servers Spielsitzungen integrieren. Siehe [Amazon GameLift ServersÜberall Flotten](#).

Authentifizierungstoken

Ein Sicherheitsnachweis, der von diesem generiert wirdAmazon GameLift Servers, authentifiziert Spieleserverprozesse, die auf Anywhere-Computerressourcen ausgeführt werden. Erforderlich,

damit Serverprozesse mit dem Amazon GameLift Servers Dienst kommunizieren können. Siehe [Amazon GameLift ServersÜberall Flotten](#).

Backend-Dienst

Eine Koordinationsebene zwischen Spielclients und dem Amazon GameLift Servers Dienst, die die Kommunikation für Anfragen zur Platzierung von Spielsitzungen steuert und Informationen zur Spielsitzung abrufen. Fungiert als sicherer Vermittler, um Hosting-Ressourcen vor böswilligen Angriffen zu schützen. Siehe [So Amazon GameLift Servers funktioniert das Hosten mit](#).

Spieleserver erstellen

Eine Paketversion deiner Gameserver-Software, die alle Dateien enthält, die für den Betrieb deines Spieleservers auf Amazon GameLift Servers Hosting-Ressourcen erforderlich sind. Enthält die ausführbare Datei, Abhängigkeiten und Ressourcen des Spieleservers. Siehe [Einen Gameserver-Build für die Bereitstellung verpacken](#).

Datenverarbeitung

Stellt in Container-Flotten eine Kopie einer Spielserver-Containergruppe auf einer Flotteninstanz dar. Jeder Computer kann jeweils eine Spielsitzung hosten. Siehe [So funktionieren Container in Amazon GameLift Servers](#).

Containerflotte

Eine Art von verwalteter Flotte, die containerisierte Spieleserver mit Container-Images anstelle herkömmlicher Builds betreibt. Bietet Flexibilität für die Bereitstellung von Spieleservern mit bestimmten Laufzeitumgebungen und Abhängigkeiten. Siehe [Amazon GameLift Serversverwaltete Containerflotten](#).

Container-Gruppe

Eine Sammlung von Containern, die zusammenarbeiten, um Spielserverfunktionen bereitzustellen. Ähnlich wie bei einer Container-Aufgabe oder einem Container-Pod werden das Verhalten von Containern, die Abhängigkeiten und die gemeinsame Nutzung von Ressourcen innerhalb einer Containerflotte definiert. Siehe [So funktionieren Container in Amazon GameLift Servers](#).

Ziel

In Warteschlangen für Spielsitzungen eine Flotte oder ein Alias, der Anfragen zur Platzierung von Spielsitzungen entgegennehmen kann. Warteschlangen können mehrere Ziele mit unterschiedlichen Prioritäten und Latenzrichtlinien haben. Siehe [Passen Sie eine Warteschlange für Spielsitzungen an](#).

Flotte

Eine Sammlung von Computerressourcen, die deine Spieleserver betreiben. Flottenressourcen können auf mehrere geografische Standorte verteilt werden, und jede Ressource führt einen oder mehrere Spielserverprozesse aus. Siehe [Stellen Sie Hosting-Flotten bereit für Amazon GameLift Servers](#).

Standort der Flotte

Eine bestimmte AWS Region oder lokale Zone, in der Flotteninstanzen bereitgestellt werden. Flotten mit mehreren Standorten können sich über mehrere Standorte erstrecken, um die Latenz zu reduzieren und die Verfügbarkeit zu verbessern. Siehe [Stellen Sie Hosting-Flotten bereit für Amazon GameLift Servers](#).

FlexMatch

Amazon GameLift ServersMatchmaking-Service, der automatisch Spielerspiele auf der Grundlage konfigurierbarer Regeln erstellt und die Platzierung der Spielsitzungen für abgeschlossene Spiele anfordert. Unterstützt kompetenzbasiertes Matching, Team-Balancing und das Auffüllen von Matches. Siehe [Was ist? GameLift FlexMatch](#)

Spielclient

Deine Spielesoftware, die auf dem Gerät eines Spielers läuft. Leitet die Platzierung von Spielern in Spielsitzungen durch Kommunikation mit einem Backend-Dienst ein und stellt für das Gameplay eine direkte Verbindung zu Spieleservern her. Siehe [So Amazon GameLift Servers funktioniert das Hosten mit](#).

Eigentum des Spiels

Schlüssel-Wert-Paare von benutzerdefinierten Daten, die einer Spielsitzung zugeordnet sind. Wird verwendet, um Konfigurationsinformationen, Spielregeln oder andere Metadaten an Spieleserver und Clients weiterzuleiten. Weitere Informationen finden Sie [GameProperty](#) in der API-Referenz.

Spieleserver

Deine benutzerdefinierte Serversoftware, die den Spielstatus verwaltet, Spieleraktionen verarbeitet und das Gameplay mehrerer verbundener Spieler synchronisiert. Lässt sich in das Amazon GameLift Servers Server-SDK integrieren, um mit dem Dienst zu kommunizieren. Siehe [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Aktivierung des Spieleservers

Der Prozess des Startens und Initialisierens eines Spieleserver-Prozesses auf einer Hosting-Ressource. Beinhaltet das Laden der ausführbaren Datei des Spieleservers, das Herstellen

der Kommunikation mit Amazon GameLift Servers und die Vorbereitung der Durchführung von Spielsitzungen. Siehe [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Container für den Spieleserver

Ein Container innerhalb einer Spielserver-Containergruppe, auf dem die Spielserver-Anwendung ausgeführt wird. Wird automatisch als unverzichtbar für den Lebenszyklus einer Containergruppe eingestuft. Siehe [So funktionieren Container in Amazon GameLift Servers](#).

Spieleserver-Containergruppe

Eine Art Containergruppe, die Container verwaltet, auf denen Spielserveranwendungen und unterstützende Software ausgeführt werden. Erforderlich für Container-Flotten zum Hosten von Spielsitzungen und kann über Flotteninstanzen hinweg repliziert werden. Siehe [So funktionieren Container in Amazon GameLift Servers](#).

Prozess auf dem Spieleserver

Eine laufende Instanz Ihres Spieleservers, die auf einer Hosting-Ressource ausgeführt wird. Jeder Prozess kann jeweils eine Spielsitzung hosten und kommuniziert direkt mit, um den Status Amazon GameLift Servers zu melden und Anweisungen zu erhalten. Siehe [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Spielsitzung

Eine Instanz Ihres Multiplayer-Spiels, die auf einem Spieleserver-Prozess ausgeführt wird. Stellt ein einzelnes Spielerlebnis dar, dem mehrere Spieler beitreten und an dem sie gemeinsam teilnehmen können. Siehe [So Amazon GameLift Servers funktioniert das Hosten mit GameSession](#) in der API-Referenz.

Daten zur Spielsitzung

Informationen im Zusammenhang mit einer Spielsitzung, darunter Spielerdaten, Spielstatus, Sitzungseigenschaften und andere Laufzeitinformationen, die von Spieleservern und Clients verwendet werden. Weitere Informationen finden Sie [GameSession](#) in der API-Referenz.

Platzierung der Spielsitzungen

Der Prozess der Suche nach verfügbaren Spieleservern zum Hosten neuer Spielsitzungen. Nutzt Echtzeitinformationen über die Verfügbarkeit von Spielservern, um anhand von Faktoren wie Kosten und Latenz optimale Platzierungsentscheidungen zu treffen. Siehe [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#) und [StartGameSessionPlacement](#) in der API-Referenz.

Schutz für Spiel-Sitzungen

Eine Einstellung, die Amazon GameLift Servers verhindert, dass Spielserverprozesse, die aktive Spielsitzungen hosten, während Skalierungsereignissen oder Flottenupdates beendet werden. Stellt sicher, dass laufende Spiele nicht unterbrochen werden. Siehe [Manuelles Einstellen der Kapazität für eine Amazon GameLift Servers Flotte](#).

Warteschlange für Spielsitzungen

Ein Platzierungsmechanismus, der konfigurierbare Algorithmen verwendet, um Spielsitzungen zu platzieren, um optimale Ergebnisse zu erzielen. Kann Platzierungen anhand von Faktoren wie den niedrigsten Hosting-Kosten und der niedrigsten Spielerlatenz an mehreren geografischen Standorten priorisieren. Siehe [Platzierung der Spielsitzungen konfigurieren](#).

Instanz im Leerlauf

Eine Flotteninstanz, die läuft, aber derzeit keine aktiven Spielsitzungen hostet. Wird als Metrik zur Überwachung der Flotteneffizienz und zur Skalierung von Entscheidungen verwendet. Siehe [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#).

Eingehende Berechtigungen

Netzwerkzugriffsregeln, die definieren, welche IP-Adressen und Portbereiche für die Verbindung mit Flotteninstanzen verwendet werden können. Steuert den Zugriff auf Spieleserver und sorgt für sichere Verbindungen. EC2 Sowohl verwaltete als auch verwaltete Containerflotten erfordern die Konfiguration von Eingangsberechtigungen.

Siehe [IpPermission](#) und [Netzwerkverbindungen konfigurieren](#).

Latenzrichtlinie

Regeln in Warteschlangen für Spielsitzungen, die akzeptable Latenzschwellenwerte für Spielerverbindungen definieren. Wird verwendet, um sicherzustellen, dass Spieler an Spielsitzungen mit akzeptabler Netzwerkleistung teilnehmen. Siehe [Passen Sie eine Warteschlange für Spielsitzungen an](#).

Verwaltete Flotte

Eine Art von Flotte, bei Amazon GameLift Servers der Rechenressourcen in der AWS Cloud bereitgestellt und verwaltet werden. Bietet automatische Skalierung, Zustandsüberwachung und Infrastrukturmanagement. Siehe [Amazon GameLift Serversverwaltete EC2 Flotten](#).

Spielersuche

Der Prozess der Gruppierung von Spielern für das Mehrspieler-Gameplay auf der Grundlage von Kriterien wie Spielstärke, Latenz oder anderen spielspezifischen Faktoren. In Amazon GameLift Servers, hauptsächlich abgewickelt von FlexMatch. Siehst du, [was ist GameLift FlexMatch?](#)

Metrikgruppe

Eine Sammlung verwandter Leistungskennzahlen, die für die Überwachung und auto-scaling von Entscheidungen verwendet werden. Gruppiert Metriken wie aktive Spielsitzungen, verfügbare Spieleserver und Spielerauslastung. Siehe [Automatische Skalierung der Flottenkapazität mit Amazon GameLift Servers](#).

Containergruppe pro Instanz

Ein optionaler Container-Gruppentyp, der zusätzliche Software auf jeder Flotteninstanz ausführt. Nützlich für Hintergrunddienste oder Überwachungstools, bei denen nur eine Kopie pro Instanz bereitgestellt wird. Siehe [So funktionieren Container in Amazon GameLift Servers](#).

Spielsitzung

Ein Verbindungssteckplatz für einen Spieler in einer Spielsitzung. Wird verwendet, um Sitzungsplätze für Spiele zu reservieren, Spieler zu überprüfen, wenn sie sich verbinden, und um den Verbindungsstatus und Verbindungsabbrüche von Spielern zu verfolgen. Siehe [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#) und [PlayerSession](#) in der API-Referenz.

Konfiguration mit Priorität

Einstellungen, mit denen die Prioritätsreihenfolge einer Warteschlange für Spielsitzungen bei der Auswahl von Zielen für die Platzierung von Spielsitzungen angepasst wird. Definiert das allgemeine Verhalten in der Warteschlange, im Gegensatz zu Prioritätslisten, bei denen die Priorität einzelner Platzierungsanfragen geändert wird. Siehe [Priorisieren Sie die Platzierung von Spielsitzungen](#).

Liste der Prioritätsüberschreibungen

Eine alternative Prioritätsliste von Orten, die in eine Platzierungsanfrage für eine Spielsitzung aufgenommen werden können. Ersetzt effektiv die in der Warteschlange konfigurierte Priorisierung für Standorte nur für diese eine Anfrage, ohne dass sich dies auf andere Anfragen auswirkt. Siehe [Priorisieren Sie die Platzierung von Spielsitzungen](#).

Prozessmanager

Eine Komponente, die den Lebenszyklus mehrerer Spielserverprozesse innerhalb eines Containers oder einer Instanz verwaltet. Übernimmt das Starten, Stoppen und Überwachen einzelner Spieleserverprozesse. Siehe [Optimiere die Laufzeitkonfiguration des Spieleservers auf verwalteten Amazon GameLift Servers](#).

Echtzeit-Skript

JavaScript Code, der die Spielelogik für Amazon GameLift Servers Echtzeitserver definiert. Verwaltet Spielerverbindungen, Nachrichtenweiterleitung und Verwaltung des Spielstatus, ohne dass ein benutzerdefinierter Spieleserver erforderlich ist. Weitere Informationen finden Sie unter [Was ist Amazon GameLift Realtime Servers?](#)

Routing-Strategie

Konfiguration, die bestimmt, wie Aliase den Verkehr an Flotten weiterleiten. Kann auf der Grundlage definierter Regeln an eine bestimmte Flotte weiterleiten oder den Verkehr auf mehrere Flotten verteilen. Siehe [Zusammenfassung einer Amazon GameLift Servers Flottenbezeichnung mit einem Alias](#).

Laufzeitkonfiguration

Einstellungen, die definieren, wie Spielserverprozesse auf Flotteninstanzen ausgeführt werden, einschließlich der Anzahl gleichzeitiger Prozesse, Startparameter und Ressourcenzuweisung. Siehe [Optimiere die Laufzeitkonfiguration des Spieleservers auf verwalteten Amazon GameLift Servers](#).

Skalierung

Der Prozess der Anpassung der Flottenkapazität durch Hinzufügen oder Entfernen von Hosting-Ressourcen auf der Grundlage der Spielernachfrage. Kann automatisch (basierend auf Nutzungskennzahlen) oder manuell (vom Administrator gesteuert) erfolgen. Siehe [Automatische Skalierung der Flottenkapazität mit Amazon GameLift Servers](#).

Richtlinie zur Skalierung

Regeln, die definieren, wann und wie die Flottenkapazität automatisch angepasst werden soll. Sie können zielbasiert (Beibehaltung einer bestimmten Auslastung) oder regelbasiert (Skalierung auf der Grundlage metrischer Schwellenwerte) sein. Siehe [Automatische Skalierung der Flottenkapazität mit Amazon GameLift Servers](#).

Server-SDK

Software-Entwicklungskit, das die Tools und Bibliotheken enthält, die für die Integration Ihres Spieleservers erforderlich sind Amazon GameLift Servers. Ermöglicht die Kommunikation zwischen deinem Spieleserver und dem Dienst für die Sitzungsverwaltung, Statusberichte und Statusaktualisierungen. Siehe [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Sitzungsbasierter Mehrspielermodus

Eine Multiplayer-Spielarchitektur, bei der das Gameplay in diskreten, zeitlich begrenzten Sitzungen stattfindet. Jede Sitzung hat einen definierten Start und ein definiertes Ende, wobei Spieler je nach Bedarf beitreten und gehen. Siehe [So Amazon GameLift Servers funktioniert das Hosten mit](#).

Spot-Instances

Eine Art von EC2 Amazon-Instance, die freie AWS Kapazität zu reduzierten Kosten nutzt. Spot-Instances können unterbrochen werden, wenn die Kapazität wieder AWS benötigt wird, sodass sie für fehlertolerante Spiele-Workloads geeignet sind. Siehe [Eine Warteschlange für Spot-Instances erstellen](#).

TLS-Zertifikat

Ein digitales Zertifikat, das für die verschlüsselte Kommunikation zwischen Spielclients und Servern verwendet wird. Erforderlich für TLS-fähige Flotten, um eine sichere Datenübertragung zu gewährleisten. Siehe [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#).

Erste Schritte mit Amazon GameLift Servers

Nutzen Sie diese Ressourcen für die ersten Schritte, um mehr über den Amazon GameLift Servers Service zu erfahren und zu erfahren, wie Sie mit der Entwicklung einer maßgeschneiderten Hosting-Lösung für Ihre sitzungsbasierten Multiplayer-Spiele beginnen können.

Bevor Sie beginnen

- Erstellen Sie eine AWS-Konto (oder geben Sie eine vorhandene an), mit der Sie es verwenden möchten. Amazon GameLift Servers
- Richten Sie Benutzer mit Berechtigungen für Amazon GameLift Servers und verwandte AWS Dienste ein.
- Wählen Sie einen aus AWS-Region , in dem Sie arbeiten möchten. Wählen Sie für die Entwicklung eine Region aus, die sich in der Nähe Ihres Standorts befindet. Sie können die Regionen jederzeit ändern.

[Richten Sie ein AWS Benutzerkonto ein](#)

Wähle deinen Weg

Ich möchte Amazon GameLift Servers schnell erkunden

Ideal für: Lernen, Erstellung eines Machbarkeitsnachweises, Rapid Prototyping

Schnellstartoptionen:

- [Gameserver-Wrapper für Amazon GameLift Servers](#) — siehe. [Tutorial: Schnelles Onboarding mit dem Wrapper Amazon GameLift Servers](#) Dieses Tool ist der schnellste und einfachste Weg, deinen Gameserver zu hosten und Spielsitzungen auszuführenAmazon GameLift Servers, ohne dass Änderungen am Spielcode erforderlich sind. Der Gameserver-Wrapper bietet grundlegende Funktionen zur Verwaltung von Spielesitzungen und eine optimierte Bereitstellung von Spieleservern. Er ist ideal für eine praktische Evaluierung der Amazon GameLift Servers Verwendung Ihres eigenen Spielprojekts oder eines Beispielprojekts. Wenn Sie bereit sind, eine benutzerdefinierte Game-Hosting-Lösung zu erstellen, wechseln Sie zu einer der benutzerdefinierten Entwicklungsoptionen mit vollständiger Integration in das Server-SDK fürAmazon GameLift Servers. Wenn dein Spiel keine benutzerdefinierte Hosting-Lösung benötigt,

kannst du den Gameserver-Wrapper weiterhin verwenden, um deine Spieleserver in der Produktion bereitzustellen und zu hosten.

- [Amazon GameLift Servers Plugin für Unreal Engine oder Unity](#) — Die Plugins bieten dir GUI-Workflows und Beispielressourcen, die dich durch die ersten Schritte führen und deinen Gameserver mit einer einfachen Hosting-Lösung bereitstellen. Verwende das Plugin, um Hosting mit selbstverwalteten Anywhere-Flotten einzurichten oder cloudbasierte, verwaltete Flotten oder EC2 Containerflotten bereitzustellen. Wenn Sie bereit sind, eine maßgeschneiderte Hosting-Lösung zu entwickeln, können Sie auf Ihren Plugin-Lösungen aufbauen.
- [Starter-Kit für Amazon GameLift Servers verwaltete Container](#) — Dieses Kit vereinfacht die Aufgaben zur Integration eines Spieleservers, zur Vorbereitung eines Gameserver-Container-Images und zur Bereitstellung einer Containerflotte für das Hosting. Für die Integration erweitert das Kit deinen Spieleserver um wichtige Funktionen zur Verwaltung von Spielsitzungen. Das Kit verwendet vorkonfigurierte Vorlagen zum Aufbau einer Containerflotte und einer automatisierten Bereitstellungs-pipeline für den Spieleserver. Wenn Sie bereit sind, vollständige Funktionen zur Verwaltung von Spielsitzungen hinzuzufügen, folgen Sie einer der benutzerdefinierten Entwicklungs-Roadmaps, für die Sie das Server-SDK integrieren möchten. Amazon GameLift Servers

Ich möchte eine maßgeschneiderte Hosting-Lösung erstellen

Ideal für: Erstellung einer Lösung auf Produktionsebene für einen benutzerdefinierten Spieleserver

Sieh dir die Seite an [Amazon GameLift Servers Hosting-Optionen für Spiele](#) und wähle eine der Optionen für deine Lösung aus. Folgen Sie einer dieser Entwicklungs-Roadmaps, um mit der Entwicklung einer maßgeschneiderten Hosting-Lösung mit vollem Funktionsumfang für Ihr Spiel zu beginnen. Die Roadmaps enthalten detaillierte Anleitungen zum Erstellen, Testen und Anpassen der einzelnen Komponenten in Ihrer Hosting-Lösung.

- [Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwaltetem EC2](#)
- [Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwalteten Containern](#)
- [Entwicklungs-Roadmap für Hosting mit Anywhere Amazon GameLift Servers](#)
- [Entwicklungs-Roadmap für Hybrid-Hosting mit Amazon GameLift Servers](#)

Empfohlener Lernpfad

1. Fangen Sie klein an. Probiere den Gameserver-Wrapper oder das Plugin für Unreal aus.

2. Verstehe die Konzepte. Überprüfe die Seite [So Amazon GameLift Servers funktioniert das Hosten mit](#).
3. Wählen Sie Architektur. Wählen Sie Ihr Hosting-Modell basierend auf den Anforderungen Ihres Spiels aus.
4. Baue und teste. Folgen Sie der entsprechenden Entwicklungs-Roadmap. Erstellen Sie für jede Komponente eine Basisversion, iterieren Sie sie und passen Sie sie an.
5. Skalieren und optimieren. Optimieren Sie Ihre Lösung für den Einsatz auf Produktionsebene. Fügen Sie erweiterte Funktionen wie Matchmaking hinzu.

Richten Sie ein AWS Benutzerkonto ein

Tip

Verwenden Sie diese Themen, um Hilfe bei diesen Aufgaben zu erhalten:

- Holen Sie sich ein neues AWS-Konto zur Verwendung mit Amazon GameLift Servers.
- Erstellen Sie einen Benutzer oder eine Gruppe mit Berechtigungen zum Arbeiten mit Amazon GameLift Servers Ressourcen.
- Richten Sie Sicherheitsanmeldedaten ein (Sie benötigen diese, um die AWS CLI Tools und Amazon GameLift Servers Plugins für Unreal und Unity zu verwenden)
-

Wie bei allen AWS Diensten benötigen Sie eine, AWS-Konto um den Amazon GameLift Servers Dienst und die Tools nutzen zu können. An AWS-Konto erfüllt zwei Hauptfunktionen: (1) Es bietet Ihnen einen Container für alle AWS Ressourcen, die Sie mit dem Konto erstellen, und (2) es ermöglicht Ihnen, die Sicherheit Ihrer AWS Ressourcen zu verwalten, einschließlich der Einrichtung der Benutzerauthentifizierung und der Kontrolle der Benutzerzugriffsberechtigungen. Die Erstellung eines ist kostenlos AWS-Konto.

Erkunden Sie Amazon GameLift Servers mit oder ohne AWS Konto

Du benötigst kein AWS Konto für:

- Entdecke AWS Tools zum Erstellen, Ausführen und Erweitern von Spielerlebnissen bei [AWS for Games](#). Lesen Sie den [Blog](#) und stöbern Sie in der [Solutions for Games-Bibliothek](#).

- Weitere Informationen finden Sie Amazon GameLift Servers in der [Produktübersicht und in den Ressourcen](#). FAQs Fragen Sie AWS nach Antworten auf Ihre Produktfragen. (Versuche es mit diesem Beispiel: „Ich suche nach kostengünstigen Optionen, um mein Multiplayer-Spiel zu hosten“.)
- Weitere Informationen finden Sie in der [technischen Dokumentation](#), einschließlich Entwicklerhandbüchern für Hosting und Matchmaking sowie dem Referenzhandbuch zur Service-API. Amazon GameLift Servers
- Informieren Sie sich über Techniken zur [Amazon GameLift ServersPreisgestaltung](#) und Kostenoptimierung. Probieren Sie den [Preisrechner](#) aus, um zu sehen, wie die Hosting-Kosten auf der Grundlage der Spitzenauslastung durch gleichzeitige Spieler (CCU) berechnet werden.
- Holen Sie sich Downloads und sehen Sie sich Code-Repositorys Amazon GameLift Servers SDKs, Plugins und Toolkits an. Siehe [Amazon GameLift ServersErste Schritte](#). (Sie benötigen eine AWS-Konto , um sie verwenden zu können.)

Sie benötigen ein AWS Konto für:

- Folgen Sie den Onboarding-Workflows mit den Amazon GameLift Servers Plugins für Unreal und Unity oder verwenden Sie den Gameserver-Wrapper.
- Erstellen und verwalten Sie AWS Ressourcen mit dem. AWS-Managementkonsole
- Erstellen und verwalten Sie AWS Ressourcen mit dem AWS Command Line Interface.
- Verwenden Sie Amazon Q mit dem In der Amazon GameLift Servers technischen Dokumentation, um Antworten, Anleitungen und Empfehlungen zu finden.

Themen

- [Melden Sie sich an für ein AWS-Konto](#)
- [Erstellen eines Benutzers mit Administratorzugriff](#)
- [Legen Sie Benutzerberechtigungen fest für Amazon GameLift Servers](#)
- [Richten Sie den programmatischen Zugriff für Benutzer ein](#)
- [Richte den programmatischen Zugriff für dein Spiel ein](#)
- [Beispiele für IAM-Berechtigungen für Amazon GameLift Servers](#)
- [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#)

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie [https://portal.aws.amazon.com/billing/die Anmeldung](https://portal.aws.amazon.com/billing/die-Anmeldung).
2. Folgen Sie den Online-Anweisungen.

Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontos wird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um [Aufgaben auszuführen, die Root-Benutzerzugriff erfordern](#).

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Du kannst jederzeit deine aktuellen Kontoaktivitäten einsehen und dein Konto verwalten, indem du zu <https://aws.amazon.com/> gehst und Mein Konto auswählst.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich [AWS-Managementkonsole](#) als Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter [Anmelden als Root-Benutzer](#) im AWS-Anmeldung -Benutzerhandbuch zu.

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

Anweisungen finden Sie unter [Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

Anweisungen finden Sie unter [Aktivieren AWS IAM Identity Center](#) im AWS IAM Identity Center - Benutzerhandbuch.

2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter [Benutzerzugriff mit der Standardeinstellung konfigurieren](#).AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie [im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal](#).

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

Anweisungen hierzu finden Sie unter [Berechtigungssatz erstellen](#) im AWS IAM Identity Center - Benutzerhandbuch.

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

Eine genaue Anleitung finden Sie unter [Gruppen hinzufügen](#) im AWS IAM Identity Center - Benutzerhandbuch.

Legen Sie Benutzerberechtigungen fest für Amazon GameLift Servers

Erstellen Sie zusätzliche Benutzer oder erweitern Sie die Zugriffsberechtigungen auf bestehende Benutzer, je nach Bedarf für Ihre Amazon GameLift Servers Ressourcen. Als bewährte Methode

([bewährte Sicherheitsmethoden in IAM](#)) sollten Sie allen Benutzern Berechtigungen mit den geringsten Rechten zuweisen. Hinweise zur Syntax von Berechtigungen finden Sie unter [Beispiele für IAM-Berechtigungen für Amazon GameLift Servers](#)

Verwenden Sie die folgenden Anweisungen, um Benutzerberechtigungen auf der Grundlage der Verwaltung der Benutzer in Ihrem AWS Konto festzulegen.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Bei der Arbeit mit IAM-Benutzern hat es sich bewährt, Berechtigungen immer Rollen oder Benutzergruppen zuzuweisen, nicht einzelnen Benutzern.

Richten Sie den programmatischen Zugriff für Benutzer ein

Benutzer benötigen programmatischen Zugriff, wenn sie mit AWS außerhalb des Interagierens möchten. AWS-Managementkonsole Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt von der Art des Benutzers ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Empfohlen) Verwenden Sie Konsolenanmeldeinformationen als temporäre Anmeldeinformationen, um programmatische Anfragen an AWS CLI AWS SDKs, oder zu signieren . AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Anmeldung für AWS lokale Entwicklung im AWS Command Line Interface Benutzerhandbuch. • Weitere Informationen finden Sie unter Anmeldung für AWS lokale Entwicklung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. AWS SDKs
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder AWS APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		SDKs und im Tools-Referenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldinformationen im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. AWS APIs

Wenn Sie Zugriffsschlüssel verwenden, finden Sie weitere Informationen unter [Bewährte Methoden für die Verwaltung von AWS Zugriffsschlüsseln](#).

Richte den programmatischen Zugriff für dein Spiel ein

Die meisten Spiele verwenden Backend-Dienste für die Kommunikation Amazon GameLift Servers mit dem. AWS SDKs Verwende einen Backend-Dienst (der für einen Spielclient agiert), um Spielsitzungen anzufordern, Spieler in Spiele einzuordnen und andere Aufgaben zu erledigen. Diese Dienste benötigen programmatischen Zugriff und Sicherheitsanmeldedaten, um Aufrufe der Service-API für zu authentifizieren. Amazon GameLift Servers

Denn Sie verwalten diesen ZugriffAmazon GameLift Servers, indem Sie einen Player-Benutzer in AWS Identity and Access Management (IAM) erstellen. Verwalten Sie die Benutzerberechtigungen von Spielern mit einer der folgenden Optionen:

- Erstellen Sie eine IAM-Rolle mit Spielerbenutzerberechtigungen und ermöglichen Sie dem Player-Benutzer, die Rolle bei Bedarf zu übernehmen. Der Back-End-Dienst muss Code enthalten, mit dem diese Rolle übernommen werden kann, bevor Anfragen gestellt werden. Amazon GameLift Servers Gemäß den bewährten Sicherheitsmethoden bieten Rollen begrenzten, temporären Zugriff. Sie können Rollen für Workloads verwenden, die auf AWS Ressourcen ([IAM-Rollen](#)) oder [außerhalb von AWS \(IAM Roles Anywhere\)](#) ausgeführt werden.
- Erstellen Sie eine IAM-Benutzergruppe mit Spielerbenutzerberechtigungen und fügen Sie Ihren Player-Benutzer der Gruppe hinzu. Mit dieser Option erhalten Ihre Player-Benutzer langfristige Anmeldeinformationen, die der Back-End-Dienst speichern und für die Kommunikation mit ihnen verwenden muss. Amazon GameLift Servers

Informationen zur Syntax der Berechtigungsrichtlinien finden Sie unter [Beispiele für Benutzerberechtigungen von Spielern](#).

Weitere Informationen zur Verwaltung von Berechtigungen für die Verwendung durch einen Workload finden Sie unter [IAM-Identitäten: Temporäre Anmeldeinformationen in IAM](#).

Beispiele für IAM-Berechtigungen für Amazon GameLift Servers

Verwenden Sie die Syntax in diesen Beispielen, um AWS Identity and Access Management (IAM-) Berechtigungen für Benutzer festzulegen, die Zugriff auf Amazon GameLift Servers Ressourcen benötigen. Weitere Informationen zur Verwaltung von Benutzerberechtigungen finden Sie unter [Legen Sie Benutzerberechtigungen fest für Amazon GameLift Servers](#). Bei der Verwaltung von

Berechtigungen für Benutzer außerhalb des IAM Identity Center hat es sich bewährt, Berechtigungen immer IAM-Rollen oder Benutzergruppen zuzuordnen, nicht einzelnen Benutzern.

Wenn Sie die Lösung Amazon GameLift Servers FleetIQ als eigenständige Lösung verwenden, finden Sie weitere Informationen unter [Einrichten Ihres AWS-Konto](#) für Amazon GameLift Servers FleetIQ

Beispiele für Administratorberechtigungen

Diese Beispiele geben einem Hosting-Administrator oder Entwickler gezielten Zugriff auf die Verwaltung von Amazon GameLift Servers Spielhosting-Ressourcen.

Example Syntax für Ressourcenberechtigungen Amazon GameLift Servers mit vollem Zugriff

Das folgende Beispiel erweitert den Vollzugriff auf alle Amazon GameLift Servers Ressourcen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "gamelift:*",
    "Resource": "*"
  }
}
```

Example Syntax für Amazon GameLift Servers Ressourcenberechtigungen mit Unterstützung für Regionen, die standardmäßig nicht aktiviert sind

Das folgende Beispiel erweitert den Zugriff auf alle Amazon GameLift Servers Ressourcen und AWS Regionen, die standardmäßig nicht aktiviert sind. Weitere Informationen zu Regionen, die standardmäßig nicht aktiviert sind, und zu deren Aktivierung finden Sie unter [Verwaltung AWS-Regionen](#) in der Allgemeine AWS-Referenz.

JSON

```
{
```

```
"Version": "2012-10-17",
"Statement": {
  "Effect": "Allow",
  "Action": [
    "ec2:DescribeRegions",
    "gamelift:*"
  ],
  "Resource": "*"
}
```

Example Syntax für Amazon GameLift Servers Ressourcen für den Zugriff auf Container-Images in Amazon ECR

Das folgende Beispiel erweitert den Zugriff auf Amazon Elastic Container Registry (Amazon ECR) -Aktionen, die Amazon GameLift Servers Benutzer benötigen, wenn sie mit verwalteten Containerflotten arbeiten.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "ecr:DescribeImages",
      "ecr:BatchGetImage",
      "ecr:GetDownloadUrlForLayer"
    ],
    "Resource": "*"
  }
}
```

Example Syntax für Amazon GameLift Servers Ressourcen und Berechtigungen **PassRole**

Das folgende Beispiel erweitert den Zugriff auf alle Amazon GameLift Servers Ressourcen und ermöglicht es einem Benutzer, eine IAM-Dienstrolle an zu Amazon GameLift Servers übergeben. Eine Servicerolle bietet Amazon GameLift Servers eingeschränkte Möglichkeiten, in Ihrem Namen auf andere Ressourcen und Dienste zuzugreifen, wie unter beschrieben [Richten Sie eine IAM-](#)

[Servicerolle ein für Amazon GameLift Servers](#). Wenn Sie beispielsweise auf eine CreateBuild Anfrage antworten, Amazon GameLift Servers benötigen Sie Zugriff auf Ihre Build-Dateien in einem Amazon S3 S3-Bucket. Weitere Informationen zu dieser PassRole Aktion finden Sie unter [IAM: Eine IAM-Rolle an einen bestimmten AWS Service weitergeben](#) im IAM-Benutzerhandbuch.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "gamelift:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "gamelift.amazonaws.com"
        }
      }
    }
  ]
}
```

Beispiele für Benutzerberechtigungen von Spielern

Diese Beispiele ermöglichen es einem Back-End-Dienst oder einer anderen Entität, API-Aufrufe an die Amazon GameLift Servers API zu tätigen. Sie behandeln die gängigen Szenarien für die Verwaltung von Spielsitzungen, Spielersitzungen und Matchmaking. Weitere Details finden Sie unter [Richte den programmatischen Zugriff für dein Spiel ein](#).

Example Syntax für die Platzierungsberechtigungen für Spielsitzungen

Das folgende Beispiel erweitert den Zugriff auf die Amazon GameLift Servers APIs , die Warteschlangen für die Platzierung von Spielsitzungen verwenden, um Spielsitzungen zu erstellen und Spielersitzungen zu verwalten.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "PlayerPermissionsForGameSessionPlacements",
    "Effect": "Allow",
    "Action": [
      "gamelift:StartGameSessionPlacement",
      "gamelift:DescribeGameSessionPlacement",
      "gamelift:StopGameSessionPlacement",
      "gamelift:CreatePlayerSession",
      "gamelift:CreatePlayerSessions",
      "gamelift:DescribeGameSessions"
    ],
    "Resource": "*"
  }
}
```

Example Syntax für Matchmaking-Berechtigungen

Das folgende Beispiel erweitert den Zugriff auf diejenigen, Amazon GameLift Servers APIs die FlexMatch Matchmaking-Aktivitäten verwalten. FlexMatchordnet Spieler für neue oder bestehende Spielsitzungen zu und leitet die Platzierung von Spielsitzungen für Spiele ein, auf denen gehostet wird. Amazon GameLift Servers Weitere Informationen zu finden Sie FlexMatch unter [Was ist Amazon GameLift Servers FlexMatch?](#)

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "PlayerPermissionsForGameSessionMatchmaking",
    "Effect": "Allow",
    "Action": [
      "gamelift:StartMatchmaking",
      "gamelift:DescribeMatchmaking",
      "gamelift:StopMatchmaking",
      "gamelift:AcceptMatch",
      "gamelift:StartMatchBackfill",

```

```
    "gamelift:DescribeGameSessions"
  ],
  "Resource": "*"
}
```

Example Syntax für die manuelle Platzierung von Spielsitzungen

Das folgende Beispiel erweitert den Zugriff auf die Amazon GameLift Servers APIs , die manuell Spielsitzungen und Spielersitzungen auf bestimmten Flotten erstellen. Dieses Szenario unterstützt Spiele, die keine Platzierungwarteschlangen verwenden, z. B. Spiele, bei denen Spieler teilnehmen können, indem sie aus einer Liste verfügbarer Spielsitzungen auswählen (die "list-and-pick" - Methode).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "PlayerPermissionsForManualGameSessions",
    "Effect": "Allow",
    "Action": [
      "gamelift:CreateGameSession",
      "gamelift:DescribeGameSessions",
      "gamelift:SearchGameSessions",
      "gamelift:CreatePlayerSession",
      "gamelift:CreatePlayerSessions",
      "gamelift:DescribePlayerSessions"
    ],
    "Resource": "*"
  }
}
```

Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers

Bei einigen Amazon GameLift Servers Funktionen müssen Sie den eingeschränkten Zugriff auf andere AWS Ressourcen, die Sie besitzen, erweitern. Sie können dies tun, indem Sie eine AWS Identity and Access Management (IAM-) Rolle erstellen. Eine IAM-[Rolle](#) ist eine IAM-Identität, die Sie in Ihrem Konto mit bestimmten Berechtigungen erstellen können. Eine IAM-Rolle ähnelt einem

IAM-Benutzer insofern, als es sich um eine AWS Identität mit Berechtigungsrichtlinien handelt, die festlegen, wofür die Identität zuständig ist und was nicht. AWS Eine Rolle ist jedoch nicht einer einzigen Person zugeordnet, sondern kann von allen Personen angenommen werden, die diese Rolle benötigen. Einer Rolle sind außerdem keine standardmäßigen, langfristigen Anmeldeinformationen (Passwörter oder Zugriffsschlüssel) zugeordnet. Wenn Sie eine Rolle übernehmen, erhalten Sie stattdessen temporäre Anmeldeinformationen für Ihre Rollensitzung.

In diesem Thema wird beschrieben, wie Sie eine Rolle erstellen, die Sie mit Ihren Amazon GameLift Servers verwalteten Flotten verwenden können. Wenn Sie Amazon GameLift Servers FleetIQ das Game-Hosting auf Ihren Amazon Elastic Compute Cloud (Amazon EC2) -Instances optimieren, finden Sie weitere Informationen unter [Einrichten Ihres AWS-Konto für Amazon GameLift Servers FleetIQ](#).

Erstellen Sie im folgenden Verfahren eine Rolle mit einer benutzerdefinierten Berechtigungsrichtlinie und einer Vertrauensrichtlinie, die es ermöglicht, die Rolle Amazon GameLift Servers zu übernehmen.

Erstellen Sie eine IAM-Servicerolle für eine Amazon GameLift Servers verwaltete EC2-Flotte

Schritt 1: Erstellen Sie eine Berechtigungsrichtlinie.

Verwenden Sie die Anweisungen und Beispiele auf dieser Seite, um eine benutzerdefinierte Berechtigungsrichtlinie für den Amazon GameLift Servers Flottentyp zu erstellen, mit dem Sie arbeiten.

So verwenden Sie den JSON-Richtlinienditor zum Erstellen einer Richtlinie

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).

Wenn Sie zum ersten Mal Policies (Richtlinien) auswählen, erscheint die Seite Welcome to Managed Policies (Willkommen bei verwalteten Richtlinien). Wählen Sie Get Started.

3. Wählen Sie oben auf der Seite Create policy (Richtlinie erstellen) aus.
4. Wählen Sie im Bereich Policy editor (Richtlinien-Editor) die Option JSON aus.
5. Geben oder fügen Sie ein JSON-Richtliniendokument ein. Weitere Informationen zur IAM-Richtliniensprache finden Sie in der [IAM-JSON-Richtlinienreferenz](#).

6. Beheben Sie alle Sicherheitswarnungen, Fehler oder allgemeinen Warnungen, die während der [Richtlinien-Validierung](#) erzeugt wurden, und wählen Sie dann Weiter.

 Note

Sie können jederzeit zwischen den Editoroptionen Visual und JSON wechseln. Wenn Sie jedoch Änderungen vornehmen oder im Visual-Editor Weiter wählen, strukturiert IAM Ihre Richtlinie möglicherweise um, um sie für den visuellen Editor zu optimieren. Weitere Informationen finden Sie unter [Richtlinienrestrukturierung](#) im IAM-Benutzerhandbuch.

7. (Optional) Wenn Sie eine Richtlinie in der erstellen oder bearbeiten AWS-Managementkonsole, können Sie eine JSON- oder YAML-Richtlinienvorlage generieren, die Sie in CloudFormation Vorlagen verwenden können.

Wählen Sie dazu im Richtlinien-Editor Aktionen und anschließend CloudFormationVorlage generieren aus. Weitere Informationen CloudFormation finden Sie in der [Referenz zum AWS Identity and Access Management Ressourcentyp](#) im AWS CloudFormation Benutzerhandbuch.

8. Wenn Sie mit dem Hinzufügen von Berechtigungen zur Richtlinie fertig sind, wählen Sie Next (Weiter) aus.
9. Geben Sie auf der Seite Prüfen und erstellen unter Richtlinienname einen Namen und unter Beschreibung (optional) eine Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie Permissions defined in this policy (In dieser Richtlinie definierte Berechtigungen), um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden.
10. (Optional) Fügen Sie der Richtlinie Metadaten hinzu, indem Sie Tags als Schlüssel-Wert-Paare anfügen. Weitere Informationen zur Verwendung von Tags in IAM finden Sie unter [Tags für AWS Identity and Access Management Ressourcen](#) im IAM-Benutzerhandbuch.
11. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.

Schritt 2: Erstellen Sie eine Rolle, die übernommen Amazon GameLift Servers werden kann.

So erstellen Sie eine IAM-Rolle

1. Klicken Sie im Navigationsbereich der IAM-Konsole auf Rollen, und wählen Sie dann Rolle erstellen.
2. Wählen Sie auf der Seite Vertrauenswürdige Entität auswählen die Option Benutzerdefinierte Vertrauensrichtlinie aus. Diese Auswahl öffnet den Editor für benutzerdefinierte Vertrauensrichtlinien.

3. Ersetzen Sie die standardmäßige JSON-Syntax durch die folgende Syntax und wählen Sie dann Weiter, um fortzufahren.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. Suchen Sie auf der Seite „Berechtigungen hinzufügen“ nach der Berechtigungsrichtlinie, die Sie in Schritt 1 erstellt haben, und wählen Sie sie aus. Wählen Sie Next (Weiter), um fortzufahren.
5. Geben Sie auf der Seite Name, überprüfen und erstellen einen Rollennamen und eine Beschreibung (optional) für die Rolle ein, die Sie erstellen. Überprüfen Sie die Vertrauensentitäten und die hinzugefügten Berechtigungen.
6. Wählen Sie Rolle erstellen, um Ihre neue Rolle zu speichern.

Erstellen Sie eine IAM-Rolle für Amazon GameLift Servers verwaltete Container

Wenn Sie Amazon GameLift Servers verwaltete Container verwenden, müssen Sie eine IAM-Servicerolle für die Verwendung mit einer Containerflotte erstellen. Diese Rolle gewährt eingeschränkte Berechtigungen, die Sie Amazon GameLift Servers benötigen, um die Ressourcen Ihrer Containerflotte zu verwalten und in Ihrem Namen Maßnahmen zu ergreifen.

Um eine IAM-Rolle für eine Containerflotte zu erstellen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>
2. Klicken Sie im Navigationsbereich der IAM-Konsole auf Rollen, und wählen Sie dann Rolle erstellen.

3. Wählen Sie auf der Seite „Vertrauenswürdige Entität auswählen“ den AWS Dienst und anschließend den Anwendungsfall "GameLift" aus. Wählen Sie Weiter
4. Wählen Sie auf „Berechtigungen hinzufügen“ die verwaltete Richtlinie `ausGameLiftContainerFleetPolicy`. Wählen Sie Weiter aus. [AWS verwaltete Richtlinien für Amazon GameLift Servers](#) Weitere Informationen zu dieser Richtlinie finden Sie unter.
5. Geben Sie unter Name, review and create einen Rollennamen ein und wählen Sie Create role aus, um die neue Rolle zu speichern.

Syntax der Berechtigungsrichtlinie

- Berechtigungen für Amazon GameLift Servers die Übernahme der Servicerolle

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Berechtigungen für den Zugriff auf AWS Regionen, die standardmäßig nicht aktiviert sind

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "gamelift.amazonaws.com",
          "gamelift.ap-east-1.amazonaws.com",

```

```
        "gamelift.me-south-1.amazonaws.com",  
        "gamelift.af-south-1.amazonaws.com",  
        "gamelift.eu-south-1.amazonaws.com"  
    ],  
    },  
    "Action": "sts:AssumeRole"  
}  
]  
}
```

Holen Sie sich Amazon GameLift Servers Entwicklungstools

Amazon GameLift Servers bietet eine Reihe von SDKs und anderen Tools, mit denen Sie Game-Hosting-Lösungen für Ihre Spiele entwickeln können. SDKs Sie erweitern Spieleserver, Spieleclients und Backend-Dienste um Funktionen, die es ihnen ermöglichen, mit dem Amazon GameLift Servers Dienst zu interagieren. Aktuelle Informationen zu Amazon GameLift Servers SDK-Versionen und Kompatibilität finden Sie unter [Amazon GameLift Servers-Versionshinweise](#).

Für Spieleserver

Integrieren und erstellen Sie Ihre 64-Bit-Spieleserver mit dem Server-SDK für Amazon GameLift Servers. Der Spieleserver verwendet das Server-SDK für die Kommunikation mit dem Amazon GameLift Servers Dienst für die Verwaltung von Spielsitzungen, einschließlich des Startens, Aktualisierens und Stoppens von Spielsitzungen. Hilfe zur Integration des Server-SDK in Ihre Spielprojekte finden Sie unter [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#).

Unterstützung bei der Entwicklung

- Entwicklungs-Betriebssystem
 - Windows
 - Linux
- Programmiersprachen

[Holen Sie sich das Amazon GameLift Servers SDK](#). Versionsspezifische Informationen und Installationsanweisungen finden Sie in den Readme-Dateien, die in jedem Paket enthalten sind.

- [C++-Server-SDK](#)
 - [Server-SDK-Referenz](#)
 - [Wie integriert man](#)

- [C#-Server-SDK](#) (Support für .NET 4, .NET 6, .NET 8 variiert je nach Version, siehe [SDK-Versionen](#))
 - [Server-SDK-Referenz](#)
 - [Wie integriert man](#)
- [Gehen Sie zum Server-SDK](#)
 - [Server-SDK-Referenz](#)
 - [Wie integriert man](#)
- Game-Engine-Unterstützung

Das Plug-in mit vollem Funktionsumfang für Amazon GameLift Servers umfasst UI-Workflows und Beispiellressourcen sowie integrierte Versionen des AWS SDK und des Server-SDK. Die Workflows führen Sie durch die Konfiguration und Bereitstellung Ihres Spieleservers für das Hosting mit verwalteten EC2-Flotten, verwalteten Container-Flotten oder selbstverwalteten Anywhere-Flotten. Wenn Sie die geführten Workflows nicht benötigen, können Sie auch einfach das Server-SDK für Ihre Game-Engine aus denselben Repositorys beziehen. GitHub

Wenn du eine andere Game-Engine oder Entwicklungsumgebung verwendest, die das Plugin nicht unterstützt, besorge dir das Server-SDK für deine Programmiersprache und füge es zu deinem Spielprojekt hinzu.

Versionspezifische Informationen und Installationsanweisungen findest du in den Readme-Dateien, die in jedem Paket enthalten sind.

- [Plugin für Unreal Engine — Entwickelt](#) für die Verwendung mit den Unreal Engine-Versionen 5.6. Suchen Sie in den versionsspezifischen Readme-Dateien nach, ob Unreal unterstützt wird.
 - [Plugin-Anleitung für Unreal Engine](#)
 - [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- [Plugin für Unity](#) — Entwickelt für die Verwendung mit den LTS-Versionen von Unity Editor 6.0, 2022.3 oder 2021.3. Es unterstützt die Profile .NET Framework und .NET Standard von Unity mit .NET Standard 2.1 und .NET 4.x. Informationen zur Unity-Unterstützung finden Sie in den versionsspezifischen Readmes.
 - [Plugin-Leitfaden für Unity](#)
 - [???SDK-Referenz für C#-Server](#)
- [Server-SDK für Unreal](#)
 - [Server-SDK-Referenz](#)

- [Amazon GameLift ServersIn ein Unreal Engine-Projekt integrieren](#)
- [Server-SDK SDK for Unity](#)
- [Server-SDK-Referenz](#)
- [Amazon GameLift ServersIn ein Unity-Projekt integrieren](#)

Runtime-Unterstützung

Für eine verwaltete Hosting-Lösung sollten Sie Ihren Spieleserver so einrichten, dass er auf einem der folgenden Amazon-Computer-Images läuft (AMIs). [Amazon GameLift ServersAMI-Versionen](#) Weitere Amazon GameLift Servers Informationen finden Sie unter.

- [Windows Server 2016](#)
- [Amazon Linux 2023](#)
- [Amazon Linux 2](#)

Note

Der Support für Amazon Linux 2 (AL2) endet am 30. Juni 2026. Weitere Einzelheiten finden Sie in [Amazon Linux FAQs 2](#). Bei Spieleservern, die auf dem Amazon GameLift Servers Server-SDK 4.x gehostet werden AL2 und dieses verwenden, aktualisieren Sie zuerst den Spieleserver-Build auf das Server-SDK 5.x und stellen Sie ihn dann auf Instances bereit AL2023 . Siehe [Migrieren Sie zum Server-SDK 5.x für Amazon GameLift Servers](#).

Zusätzliche Tools

[Spieleserver-Wrapper für Amazon GameLift Servers](#)

Dieses Tool hilft Ihnen bei der Bereitstellung eines Spieleservers zum Hosten mit einer Reihe grundlegender Funktionen zur Verwaltung von Spielesitzungen. Mit diesem Tool müssen Sie keine Änderungen an Ihrem Spielcode vornehmen oder das Server-SDK für integrierenAmazon GameLift Servers. Verwenden Sie den Gameserver-Wrapper, um Ihren Spieleserver zu verpacken und ihn mithilfe einer der drei Hosting-Lösungen (Anywhere, Managed EC2 oder Managed Containers) für das Amazon GameLift Servers Game-Hosting bereitzustellen. Dieses Tool eignet sich am besten für die frühe Evaluierung oder das Prototyping mit Ihrem eigenen Spiel oder einem Beispielspiel, da dieses Tool die Anpassung von Spieleservern nicht unterstützt. Wenn dein Spiel

keine benutzerdefinierten Funktionen benötigt, kannst du deinen Spielservers mit dem Gameserver-Wrapper für das Produktionshosting bereitstellen.

[Amazon GameLift Servers Toolkit](#)

Das Amazon GameLift Servers Toolkit ist eine Sammlung von Skripten und anderen Tools, die wir entwickelt haben, um Entwicklern bei häufigen Szenarien und Problemen zu helfen. Zu den Materialien des Toolkits gehören Skripte, Beispielcode und Readmes.

- [Container-Starterkit](#) — Verwenden Sie dieses Tool, um die Einrichtung von Spielservers-Builds für das Hosting mit Amazon GameLift Servers verwalteten Containern zu vereinfachen. Das Kit integriert wichtige Funktionen zur Verwaltung von Spielsitzungen in einen Spielservers und verwendet vorkonfigurierte Vorlagen, um eine Containerflotte zu erstellen und eine automatisierte Bereitstellungspipeline für Ihren Spielservers-Build einzurichten. Nach der Bereitstellung kannst du mithilfe der Amazon GameLift Servers Konsole und der API-Tools die Leistung deiner Flotte überwachen, Spielsitzungen verwalten und Messwerte analysieren. Das Kit lässt sich AWS CodeBuild für die Build-Automatisierung, Amazon Simple Storage Service für die Speicherung und AWS CloudFormation für die Infrastrukturbereitstellung integrieren.
- [Tool zur schnellen Aktualisierung von Builds](#) — Verwenden Sie dieses Tool, um einen Spielservers-Build zu ändern, der bereits auf einer verwalteten EC2-Flotte bereitgestellt wurde. Das Tool wurde entwickelt, um dir dabei zu helfen, Spiele-Build-Dateien schnell auszutauschen, ohne bei jeder Änderung neue EC2-Flotten konfigurieren und erstellen zu müssen. Sie können einzelne Instances oder alle Instances in der Flotte aktualisieren. Mit den Optionen kannst du bestimmte Build-Dateien oder einen ganzen Build ersetzen und festlegen, wie du Spielservers nach den Updates neu startest.

Für Spieleclients

Integrieren Sie Ihren Spieleclient in das Client-SDK für Amazon GameLift Servers. Das Client-SDK bietet Dienstprogramme für Spieleclients, mit denen sie Amazon GameLift Servers Funktionen wie Player-Gateway und UDP-Ping-Beacons nutzen können.

Holen Sie sich das Client-SDK

- [Client-SDK SDK for C++](#) — Enthält Player-Gateway- und UDP-Ping-Beacon-Dienstprogramme mit integrierten Algorithmen zur Endpunktauswahl sowie eine Beispielanwendung, die die Integration demonstriert. Erfordert C++17 und eine Threading-Bibliothek.

- [Client-SDK für Unreal Engine](#) — Beinhaltet Player-Gateway (mit oder ohne Engine-Änderungen) und UDP-Ping-Beacons mit integrierten Algorithmen zur Endpunktauswahl. Erfordert die im Quellcode erstellte Unreal Engine 5.1 oder höher. Für den Zugriff ist eine Mitgliedschaft in der [Epic Games-Organisation](#) am erforderlich. GitHub

Anleitungen zur Integration findest du in der Dokumentation zu den einzelnen GitHub Repositories.

Für Backend-Dienste für Spiele

Erstellen Sie einen 64-Bit-Backend-Dienst für Ihr Spiel und integrieren Sie ihn in Funktionen aus dem AWS SDK, zu dem auch die Service-API für gehört. Amazon GameLift Servers Verwenden Sie den Backend-Service, um Interaktionen mit dem Amazon GameLift Servers Dienst abzuwickeln, einschließlich des Startens oder Findens von Spielsitzungen und des Beitritts von Spielern zu Spielen.

[Holen Sie sich das SDK AWS](#)

Weitere Informationen zur Verwendung des AWS SDK mit Amazon GameLift Servers finden Sie in den folgenden Ressourcen:

- [Amazon GameLift Servers API Referenz](#)
- Integrieren Sie die Funktionalität des Spielclients
 - [Clientseitige Serviceintegration für Spiele mit benutzerdefinierten Spieleserver-Builds](#)
 - [Clientseitige Serviceintegration für Spiele mit Servern Realtime](#)
- [Entwerfen Sie einen Backend-Service](#)

Für das Amazon GameLift Servers Ressourcenmanagement

Verwenden Sie die folgenden Tools, um Ihre Amazon GameLift Servers verwalteten Hosting-Ressourcen zu erstellen, zu aktualisieren und zu überwachen.

- [AWS-Managementkonsole](#)— Die AWS Konsole ist eine webbasierte Anwendung, die einen zentralen Zugriff auf alle einzelnen AWS Servicekonsolen bietet, einschließlich Amazon GameLift Servers. Verwenden Sie die Konsole, um ein AWS Konto zu erstellen oder sich anzumelden, und öffnen Sie die Amazon GameLift Servers Konsole, um mit Ihren Spielhosting-Ressourcen zu arbeiten. Du kannst Hosting-Flotten und andere Ressourcen konfigurieren und bereitstellen,

Nutzungs- und Leistungskennzahlen einsehen, Ressourcen im Dashboard verfolgen und viele andere Aufgaben erledigen. [Gehen Sie zur Amazon GameLift Servers Konsole.](#)

- [Service-API für Amazon GameLift Servers](#) — Diese API ermöglicht Ihnen programmatischen Zugriff auf all Ihre Amazon GameLift Servers Ressourcen. Sie ist Teil des AWS SDK, das Sie für die Verwendung mit den gängigsten Programmiersprachen herunterladen können. [Holen Sie sich das AWS SDK.](#)
- [AWS Befehlszeilenschnittstelle \(CLI\)](#) — Mit der AWS CLI können Sie über eine Befehlszeilen-Shell mit AWS Diensten interagieren. Die Tools bieten direkten Zugriff auf AWS Dienste sowie auf benutzerdefinierte Befehle, die für einen Dienst APIs verfügbar sind. [Holen Sie sich die AWS CLI.](#)
- [AWS CloudFormation](#) für Amazon GameLift Servers — Der CloudFormation Service unterstützt Sie bei der Modellierung und Einrichtung von AWS Ressourcen, um die Bereitstellung und Verwaltung der Infrastruktur zu optimieren. Erstellen Sie eine CloudFormation Vorlage, um die Amazon GameLift Servers Ressourcen für Ihre Hosting-Lösung zu beschreiben, und verwenden Sie diese Vorlage dann, um zusätzliche Ressourcen zu erstellen oder Konfigurationen zu aktualisieren. Sehen Sie sich die [Referenz zum Amazon GameLift Servers Ressourcentyp an.](#)

Für Amazon GameLift Servers Realtime

Konfigurieren und implementieren Sie Realtime Server, um Ihre Multiplayer-Spiele zu hosten. Verwenden Sie das Amazon GameLift Servers Realtime Client-SDK, damit Ihre Spieleclients eine Verbindung zu Realtime Servern herstellen können. Laden Sie zunächst [das Realtime Client-SDK herunter](#). Informationen zur Konfiguration finden Sie unter [Integrieren eines Spieleclients für Amazon GameLift Servers Realtime](#).

SDK -Unterstützung

Das Realtime Client SDK enthält die Quelle für die folgenden Sprachen:

- C# (.NET)

Entwicklungsumgebungen

Erstellen Sie das SDK nach Bedarf aus dem Quellcode für die folgenden unterstützten Entwicklungsbetriebssysteme und Game-Engines:

- Betriebssysteme — Windows, Linux, Android, iOS
- Spiele-Engines — Unity, Engines, die C#-Bibliotheken unterstützen

Betriebssysteme für Spiel-Server

Sie können Realtime Server auf Hosting-Ressourcen bereitstellen, die auf den folgenden Plattformen ausgeführt werden:

- [Amazon Linux 2](#)

Note

AL2 nähert sich dem Ende des Supports. Weitere Einzelheiten finden Sie in [Amazon Linux FAQs 2.](#)

- [Amazon Linux 2023](#)

Tutorial: Schnelles Onboarding mit dem Wrapper Amazon GameLift Servers

Willkommen zum Onboarding-Tutorial für Amazon GameLift Servers. In diesem Tutorial stellen Sie Ihren Spieleserver schnell bereit, sodass er auf einer Flotte von Cloud-basierten Rechenressourcen gehostet wird. Verwenden Sie dieses Tutorial, um die Arbeit der Integration des Server-SDK für Amazon GameLift Servers in Ihren Spielcode zu überspringen und stattdessen Ihr Spiel mit den minimalen Funktionen bereitzustellen, um mit dem Amazon GameLift Servers Dienst zu kommunizieren und Spielsitzungen durchzuführen. Du richtest eine grundlegende Hosting-Lösung ein und nutzt sie, um alle Funktionen wie automatische Skalierung und Spielersuche zu nutzen. Es ist auch eine großartige Möglichkeit, einen Prototyp Ihres Spiels als Teil einer Live-Demo oder zum Testen zu hosten.

Die wichtigsten Vorteile dieser Onboarding-Methode:

- Stellen Sie Ihren Spieleserver schnell bereit, um ihn schnell zu hosten.
- Keine Änderungen am Spielcode und keine Änderungen erforderlich.
- Verwenden Sie diese Methode mit jeder ausführbaren Datei eines Spiels, unabhängig von der Spiel-Engine.
- Erkunden Sie alle Amazon GameLift Servers Verwaltungstools, einschließlich der Überwachung der Aktivität von Spielsitzungen und des Zustands des Hostings.

 Note

Der Wrapper ist für Evaluierungszwecke und grundlegende Produktionszwecke vorgesehen. Erweiterte Funktionen wie die detaillierte Verwaltung von Spielsitzungen erfordern eine vollständige Server-SDK-Integration.

Voraussetzungen

Stellen Sie vor Beginn sicher, dass Sie über Folgendes verfügen:

- Und AWS-Konto mit den entsprechenden Berechtigungen
- AWS CLI installiert
- Gehe zu 1.18+
- Ein ausführbarer Multiplayer-Gameserver
- Marke (Linux/Mac)
- Git wurde mit einem aktiven Konto installiert

-Übersicht

In diesem Tutorial wirst du:

1. Holen Sie sich den Wrapper und bauen Sie ihn
2. Bereite den Spiel-Build vor
3. Konfiguriere den Wrapper
4. Lade den Spielserver-Build hoch
5. Erstelle die verwaltete EC2 Flotte
6. Erstelle eine Spielsitzung und stelle eine Verbindung zu ihr her
7. Überwachen und verwalten Sie Ihre Spieleserver
8. Skalieren Sie Ihre Spieleserver

Schritt 1: Holen Sie sich den Gameserver-Wrapper und bauen Sie ihn

Verwenden Sie die folgenden Befehle, um die Quelle des Gameserver-Wrappers abzurufen und den Wrapper zu erstellen. Diese Befehle verwenden SSH, aber du kannst auch direkt zum Github-Repository gehen.

Windows

```
> git clone git@github.com:amazon-gamelift/amazon-gamelift-servers-game-server-wrapper.git
> cd amazon-gamelift-servers-game-server-wrapper
> powershell -file .\build.ps1
```

Mac und Linux

```
$ git clone git@github.com:amazon-gamelift/amazon-gamelift-servers-game-server-wrapper.git
$ cd amazon-gamelift-servers-game-server-wrapper
$ make
```

Bei einem erfolgreichen Build wird ein Out-Verzeichnis hinzugefügt `amazon-gamelift-servers-game-server-wrapper`. In diesem Verzeichnis befinden sich drei Ordner, einer für jede unterstützte Hosting-Flottenoption, die eine Reihe von Build-Artefakten enthalten. In diesem Tutorial führen Sie die Bereitstellung auf verwalteten EC2 Flotten durch, sodass Sie den Ordner `gamelift-servers-managed-ec2` verwenden.

Schritt 2: Bereite deinen Gameserver-Build vor

In diesem Schritt bereiten Sie die Build-Dateien Ihres Spieleservers für das Hochladen auf den Amazon GameLift Servers vor.

Erstelle das Spielverzeichnis

Bereite jetzt ein Spielverzeichnis auf deinem lokalen Computer vor. Dieses Verzeichnis muss alle Dateien enthalten, mit denen du deinen Spieleserver ausführen musst Amazon GameLift Servers.

Dazu gehören der Gameserver-Wrapper, dein Gameserver-Build und die Datei `config.yaml`, die dafür sorgt, dass der Wrapper mit deinem Gameserver funktioniert.

Gehen Sie dazu wie folgt vor:

1. **Verwaltete Flotte. EC2** Suchen Sie im Wrapper-Ausgabeordner des Gameservers nach dem Build-Artefakt, das Sie in einer verwalteten EC2 Flotte einsetzen können. Der Build wird in ein Ausgangsverzeichnis geschrieben, wie hier angegeben: `out\linux\amd64\gamelift-servers-managed-ec2`
2. Kopieren Sie die ausführbare Datei Ihres Spieleservers und alle zugehörigen Dateien, die zum Ausführen erforderlich sind, in den `gamelift-servers-managed-ec2` Ordner. Du kannst nach Bedarf verschachtelte Verzeichnisse einrichten.

Ein Beispiel für eine Verzeichnisstruktur sieht wie folgt aus:

```
gamelift-servers-managed-ec2
#-- config.yaml
#-- amazon-gamelift-servers-game-server-wrapper
#-- MyGame
# #-- server-executable.exe
# #-- my-game-settings
# # .....
```

Schritt 3: Konfigurieren Sie den Wrapper für Ihre Flotte

Amazon GameLift Servers verwaltet den Lebenszyklus der Recheninstanzen einer Flotte, indem neue Instances mit Ihrem installierten Server-Build eingerichtet und Instances nach Bedarf recycelt werden. Der Dienst verwaltet den Prozesslebenszyklus des Spieleservers, der auf jeder Instanz ausgeführt wird. Eine verwaltete EC2 Flotte kann Instanzen an mehreren Standorten haben, um Spieler zu unterstützen, wo auch immer sie sich befinden.

Bearbeiten Sie die Datei `config.yaml`, um den Wrapper für die Protokollierung, Porteinrichtung und Serverinitialisierung zu konfigurieren.

1. Konfigurieren Sie die Logging-Einstellungen. Der Gameserver-Wrapper generiert Protokollnachrichten für jeden Gameserver-Prozess. Standardmäßig ist die Protokollebene auf Debug eingestellt, um maximale Ausführlichkeit zu gewährleisten. Dies ist bei der Einrichtung und

- bei der Problembehandlung sehr nützlich und bestimmt, wie detailliert die Protokollmeldungen sind — in diesem Fall die ausführlichsten. Zu den Optionen gehören Debug, Info, Warn und Error (am wenigsten ausführlich).
2. Geben Sie den Pfad zum Protokollverzeichnis des Spieleservers an. Der Standardpfad für Spieleserver-Logs ist `./game-server-logs`. Dieses Verzeichnis enthält alle Protokolle, die von deinem Spieleserver generiert wurden, und jede Instanz hat sie. Die Protokolle werden automatisch dorthin hochgeladen, Amazon GameLift Servers wo sie über den Tab „Ereignisse“ abgerufen werden können. Weitere Informationen finden Sie im Abschnitt Fehlerbehebung.
 3. Definieren Sie die Netzwerk-Port-Konfiguration. Stellen Sie den Gameport auf den von Ihnen bevorzugten Wert ein. Geben Sie für dieses Tutorial nur einen Port an, da Sie eine Flotte erstellen, auf der nur ein Gameserver-Prozess gleichzeitig pro Instanz ausgeführt wird. Wenn du dich dafür entscheidest, mehrere Prozesse gleichzeitig auszuführen, musst du genügend Ports für jeden gleichzeitigen Prozess konfigurieren. Der Standardwert ist 37016, wie in der Konfigurationsdatei gezeigt, aber im Allgemeinen sollten Sie für Flotten, die Linux-Builds verwenden, die Ports 22 und 1026-60000 verwenden. Verwenden Sie für Flotten, die Windows-Builds verwenden, die Ports 1026-60000.
 4. Richte den Pfad zur ausführbaren Datei des Spieleservers ein. `./MyGame/my-server-executable.exe` Passen Sie dazu den Pfad zur ausführbaren Datei Ihres Spieleservers mit dem tatsächlichen Namen und Speicherort an. Dies ist der Einstiegspunkt für den Start deines Gameservers.
 5. Konfiguriere die Gameserver-Argumente. Geben Sie mindestens das Argument `-port` an und verwenden Sie denselben Game-Port-Wert, den Sie zuvor definiert haben. Der „pos“-Wert 0 gibt an, dass dies das erste Argument ist. Fügen Sie nach Bedarf weitere Argumente hinzu. Diese Argumente werden beim Start an deinen Spieleserver übergeben, sodass du sein Laufzeitverhalten konfigurieren kannst.
 - a. Argument: `--port`
 - b. Wert: `"{{.port number here}}"`
 - c. Position: 0 (Erstes Argument in der Liste)

Beispielkonfiguration:

```
log-config:
  wrapper-log-level: debug
game-server-logs-dir: ./game-server-logs
ports:
```

```
gamePort: 37016
game-server-details:
  executable-file-path: ./MyGame/my-server-executable
game-server-args:
  - arg: "--port"
    val: "{{.gameport}}"
    pos: 0
```

Schritt 4: Lade den Spielserver-Build hoch

Du hast jetzt alle erforderlichen Elemente deines Gameserver-Builds abgeschlossen (Gameserver-Wrapper, config.yaml und deine Gameserverdateien) und bist bereit, deinen Game-Build zum Hosten hochzuladen. Amazon GameLift Servers Am schnellsten kannst du deinen Game-Build hochladen, indem du den AWS CLI Befehl verwendest, `upload-build` wie im folgenden Beispiel gezeigt.

Einen Game-Build mit Windows hochladen:

```
aws gamelift upload-build \
  --name gamelift-test-2025-03-11-1 \
  --build-version gamelift-test-2025-03-11-1 \
  --build-root out/windows/amd64/gamelift-servers-managed-ec2 \
  --operating-system WINDOWS_2022 \
  --server-sdk-version 5.3.0 \
  --region us-west-2
```

Note

Verwenden Sie für Mac- und Linux-Builds `--operating-system AMAZON_LINUX_2023`

Wenn Sie den Build erstellen, notieren Sie sich die Build-ID aus der API-Antwort, um sie für die Flottenerstellung zu verwenden.

Schritt 5: Erstellen Sie die verwaltete EC2 Flotte

Die folgenden Schritte beschreiben eine minimale Flottenkonfiguration, damit Sie so schnell wie möglich einsatzbereit sind.

So erstellen Sie Ihre Flotte:

1. Loggen Sie sich in die ein AWS-Managementkonsole und navigieren Sie zu Amazon GameLift Servers.
2. Vergewissere dich in der Menüleiste oben im Konsolenfenster, in welcher Region sich dein Build befindet. Notiere dir das, denn deine Flotte muss sich in derselben Region befinden, sonst kannst du deinen Build nicht finden oder auswählen.
3. Wählen Sie im Navigationsbereich im EC2 Abschnitt Verwalte die Option Builds aus.
4. Wählen Sie den Build aus, den Sie zuvor hochgeladen haben, um die Seite mit den Build-Details anzuzeigen.
5. Wählen Sie im Bereich „Flotten“ die Option „Flotte erstellen“. Daraufhin wird die Seite „Verwaltete EC2 Flotte definieren“ angezeigt, auf der Sie den Status Ihrer Flotte verfolgen und auf der Registerkarte „Ereignisse“ Ereignisse bei der Flottenerstellung einsehen können.
6. Geben Sie den Namen und die Beschreibung ein und wählen Sie Weiter.
7. Auf der Seite Instanzdetails definieren wird standardmäßig die Region des Builds angezeigt. Wählen Sie alle weiteren Regionen aus, die Sie hinzufügen möchten.
8. Wählen Sie als Flottenart die Option On-Demand.
9. Wählen Sie unter Instanztypen die Option c5.large und dann Weiter aus.
10. Unter Runtime-Konfiguration müssen Sie, da Ihr hochgeladenes Spiel-Build den Wrapper verwendet, stattdessen auf die ausführbare Wrapper-Datei verweisen. Für Windows-Spieleserver ist das `C:\game\amazon-gamelift-servers-game-server-wrapper.exe`. Für Linux-Spieleserver ist das `/local/game/amazon-gamelift-servers-game-server-wrapper`.

Beispiel: `LaunchPath": "/local/game/amazon-gamelift-servers-game-server-wrapper", "ConcurrentExecutions": 1, "Parameters": "-port 37016`

Konfigurieren Sie außerdem die Port-Werte des Spiels mit einem Bereich, der den Port berücksichtigt, der in den Startparametern `config.yaml` und den Laufzeitkonfigurationsparametern festgelegt wurde. Der Port in muss `config.yaml` nicht mit dem Port übereinstimmen, der in der Laufzeitkonfiguration der Flotte angegeben ist, aber wenn die Konfiguration zur Laufzeit einen anderen Port angibt, überschreibt dieser Wert den `config.yaml` Wert in. Die in der Laufzeitkonfiguration eingegebenen Startparameter überschreiben auch die Angaben in der `config.yaml`.

11. Überprüfen Sie auf der Seite „Überprüfen und erstellen“ alle Konfigurationen und wählen Sie dann „Senden“, um Ihre Flotte zu erstellen. Der Flottenstatus ändert sich, wenn die Kapazität

zum Hosten des Spielerservers erhöht wird. Sehr bald wird der Status Aktiv angezeigt. Sobald die Aktivierung abgeschlossen und die Flotte bereitgestellt ist, startet der Dienst den Wrapper, der bereit ist, eine Anfrage für eine Spielsitzung zu empfangen.

Schritt 6: Erstelle eine Spielsitzung und stelle eine Verbindung zu ihr her

Wenn dein Flottenstatus Aktiv anzeigt, bedeutet das, dass die Spieleserver bereit sind und darauf warten, eine Spielsitzung zu veranstalten. Um eine Spielsitzung zu starten, stellen Sie eine Anfrage für eine Spielsitzung an den Amazon GameLift Servers Dienst. Hier verwenden Sie den, AWS CLI um diese Anfrage zu stellen.

Note

Denken Sie daran, dass das Erstellen einer Spielsitzung mit dem nützlich AWS CLI ist, um den Prozess zu testen und sich mit ihm vertraut zu machen. Irgendwann werden Sie programmatische AWS SDK-Aufrufe zu Ihrem Spiel-Backend-Service hinzufügen, als Teil des Matchmaking- oder Platzierungssystems für Spielsitzungen Ihres Spiels.

Verwenden Sie Folgendes, um eine Spielsitzung zu erstellen:

```
aws gamelift create-game-session \  
--fleet-id <FLEET_ID> \  
--game-properties '[{"Key": "exampleProperty", "Value": "exampleValue"}]' \  
--maximum-player-session-count 3 \  
--region us-west-2
```

Sie können auch benutzerdefinierte Spieleigenschaften an Ihre ausführbare Serverdatei übergeben. Weitere Informationen finden Sie in der README-Datei unter Gameserver-Argumente. Wenn er den create-game-session Aufruf erhält, Amazon GameLift Servers informieren sie den Wrapper, die ausführbare Datei des Spieleservers zu starten und eine Spielsitzung zu starten. Was darin enthalten ist, config.yaml wirkt sich auf die Konfiguration des Spielerservers aus, und die Startparameter, die in der Konsole festgelegt sind, bestimmen die Konfiguration der Spielsitzung selbst.

Beispielformat für das Hinzufügen von Spieleigenschaften:

```
defaultArgs:
  - arg: "--port"
    val: "{{.GamePort}}"
    pos: 0
  - arg: "--ipAddress"
    val: "{{.IpAddress}}"
    pos: 1
  - arg: "--gameSessionId"
    val: "{{.GameSessionId}}"
    pos: 2
```

Schritt 7: Verwalte und überwache deine Flotte

Nun, da deine Spieleserver-Flotte eingerichtet und eine Spielsitzung gestartet wurde, kannst du sie von der Amazon GameLift Servers Konsole aus verwalten und überwachen. Das geht am besten über die Seite mit den Flottendetails, auf der du die Flottendetails bearbeiten oder die Flottenskalierung und Kapazität auf der Registerkarte Skalierung ändern kannst. Im folgenden Abschnitt findest du Informationen zur Skalierung deiner Spieleserver.

Wähle den Tab Metriken, um dir Diagramme anzusehen, die die Überwachung von Aktivitäts- und Hardwaremetriken darstellen. Weitere Informationen zu den Metrikdiagrammen erhalten Sie, wenn Sie auf den Link Info neben der ID Ihrer Flotte klicken. Zusätzlich kannst du deine Spieleserver über den Tab „Metriken“ genau überwachen, aber du kannst diesen Metriken im CloudWatch Dashboard auch Alarmer hinzufügen.

So navigierst du von der Konsole aus zum CloudWatch Dashboard:

1. Geben Sie "CloudWatch" in die Suchleiste ein und wählen Sie es aus der Suchergebnisliste aus, um die CloudWatch Übersicht anzuzeigen.
2. Scrollen Sie nach unten und wählen Sie „GameLift Dashboard anzeigen“, um Grafiken zu sehen, die auf den wichtigsten Kennzahlen Ihrer Flotten und Spielsitzungen basieren.

Schritt 8: Skalieren Sie Ihre Spieleserver

Im nächsten Schritt richten Sie Auto Scaling ein. Mit Auto Scaling skalieren Sie Ihre Flottenkapazität dynamisch als Reaktion auf die Aktivität des Spieleservers. Wenn Spieler ankommen und Spielsitzungen starten, fügt Auto Scaling weitere Instanzen hinzu, und wenn die Nachfrage der Spieler sinkt, werden durch Auto Scaling ungenutzte Instanzen freigegeben. Dies ist eine effektive

Methode, um Ihre Hosting-Ressourcen und -Kosten zu minimieren und gleichzeitig ein reibungsloses, schnelles Spielerlebnis zu bieten.

Während du dich auf den Start des Spiels vorbereitest, solltest du die auto Skalierung für deine Flotten einrichten. Auto Scaling wird als effektive Methode empfohlen, um deine Hosting-Ressourcen und -Kosten zu minimieren und gleichzeitig ein reibungsloses, schnelles Spielerlebnis zu bieten.

Um die Flottenkapazität manuell einzurichten

Um die Flottenkapazität manuell einzurichten

1. Gehen Sie auf der Detailseite Ihrer Flotte zur Registerkarte Skalierung.
2. Wählen Sie einen Standort aus und klicken Sie auf Bearbeiten.
3. Ändern Sie den Wert „Gewünschte Instanzen“ und ändern Sie die Einstellungen für „Min.“ und „Max. Größe“, um über die aktuellen Werte hinaus zu skalieren, und wählen Sie „Bestätigen“.

Note

Verwenden Sie die Einstellung „Max. Instanzen“ als Notlösung, um übermäßige Skalierung und Kosten zu vermeiden.

So verwenden Sie zielbasiertes Auto Scaling

So verwenden Sie zielbasiertes Auto Scaling

Zielbasiertes Autoscaling (Target Tracking) verknüpft die Flottenskalierung mit dem Prozentsatz der verfügbaren Spielsitzungen. Wenn Spieler dein Spiel schnell spielen und die Anzahl der verfügbaren Spielsitzungen abnimmt, reagiert das System, indem es automatisch weitere Instanzen zur Flotte hinzufügt.

1. Wählen Sie unter Zielbasierte Auto-Scaling-Richtlinie die Option Richtlinie hinzufügen aus und legen Sie fest, dass sich die Kapazität der Flotte automatisch ändert, wenn sie den von Ihnen festgelegten Schwellenwert für den Prozentsatz verfügbarer Spielsitzungen erreicht. Ein größerer Puffer kann Überspannungen besser bewältigen und sorgt dafür, dass neue Spieler schnell mit Spielen beginnen, aber das kann auch zu höheren Hosting-Kosten führen.
2. Wählen Sie Bestätigen, um die Änderungen zu akzeptieren.

Regelbasierte auto Skalierung bietet Ihnen eine detailliertere Steuerung, z. B. die Möglichkeit, die Skalierung mit anderen Flottenkennzahlen zu verknüpfen und benutzerdefinierte Schwellenwerte und Skalierungsreaktionen festzulegen. Es bietet leistungsstarke Optionen, erfordert aber auch die Verwendung der CLI und umfangreiche Tests, um zu verstehen, wie sich benutzerdefinierte Regeln in Aktion verhalten. Dieses Tutorial konzentriert sich auf die erste Einrichtung des zielbasierten Ansatzes.

Beheben Sie häufig auftretende Probleme

Im Folgenden finden Sie häufig auftretende Probleme, die beim Ausführen Ihrer Spieleserver und Spielsitzungen auftreten können. Wenn entweder der Server oder die Spielsitzung nicht richtig laufen, besteht der erste Schritt darin, die Protokolle zu überprüfen, die eines der unten beschriebenen Probleme bei neuen Bereitstellungen oder Spielen in der Produktion aufdecken können.

Folgendes wird häufig in den Protokollen aufgedeckt:

- Der Spieleserver-Prozess kann nicht gestartet werden. Dies könnte ein Fehler in der Wrapper-Konfiguration sein. Vergewissern Sie sich, dass die Datei den richtigen Startpfad und die richtigen Startparameter und Argumente hat.
- Der Spielserver-Build kann nicht ausgeführt werden. Dies ist wahrscheinlich ein Fehler im Spielcode.
- Spieler können keine Verbindung zu Spielsitzungen herstellen. Dies ist wahrscheinlich ein Fehler bei der Portkonfiguration.
- Verzögerte oder langsame Verbindungen. Überprüfen Sie die Skalierungsrichtlinien und Schwellenwerte.
- Keine Verbindung. Überprüfen Sie die Portregeln und die Konfiguration für Ihre Flotte.

Um die Ereignisprotokolle für Ihre Amazon GameLift Servers Flotte einzusehen

Um die Ereignisprotokolle für Ihre Amazon GameLift Servers Flotte einzusehen

1. Öffnen Sie die Amazon GameLift Servers-Konsole.
2. Wählen Sie auf der Seite mit den Flottendetails die Registerkarte Ereignisse und laden Sie das Protokoll herunter. Auf dem Tab „Metriken“ kannst du auch Aktivitäten und Hardwaremetriken für den Zustand des Spieleservers und die Aktivierung von Spielsitzungen überwachen.

Um Spielsitzungsprotokolle einzusehen

Um Spielsitzungsprotokolle einzusehen

1. Öffne von der Konsole aus deine Flotte und öffne den Tab Spielsitzungen.
2. Wähle eine Spielsitzungs-ID aus der Liste aus, um die entsprechende Übersichtsseite aufzurufen.
3. Wähle Protokolle herunterladen, um eine Protokolldatei lokal herunterzuladen.

Verwenden Sie die `GetGameSessionLogURL` API, um Spielsitzungsprotokolle mit der CLI anzuzeigen. Amazon GameLift Servers speichert die Protokolle automatisch für 14 Tage.

Sie können auch CloudWatch Amazon-Logs für Ihre Flotte konfigurieren. Dies bietet zusätzliche Protokollierungsfunktionen und die Integration mit anderen AWS Überwachungsdiensten.

Für Protokollzugriff in Echtzeit oder längere Aufbewahrungsfristen durch CloudWatch:

1. Geben Sie oben im Amazon GameLift Servers Konsolen-Dashboard "CloudWatch" in die Suchleiste ein und wählen Sie es aus der Ergebnisliste aus.
2. Gehen Sie zu den CloudWatch Protokollgruppen und suchen Sie nach bestimmten Sitzungen. Am einfachsten ist es, auf Alle suchen zu klicken und mit `gameSessionId` unserer `clientId` zu filtern.

Nächste Schritte

- [Erstellen Sie eine Flotte mit mehreren Standorten, um Hosting an weiteren Standorten hinzuzufügen](#)
- [Füge eine Warteschlange für Spielsitzungen hinzu, um die bestmöglichen Platzierungen für Spielsitzungen an mehreren Standorten zu erreichen](#)
- [Experimentiere mit der FlexMatch Spielerzuweisung, indem du einen Matchmaker und einen Regelsatz für dein Spiel erstellst](#)
- [Fangen Sie an, an der Funktionalität Ihres Spielclients und der Backend-Servicekomponenten zu arbeiten, damit Spieler Beitrittsanfragen stellen und sich direkt mit Spielsitzungen verbinden können](#)
- [Wenn Sie bereit sind, wechseln Sie zu einer vollständig integrierten Lösung](#)

Beispiele für Amazon GameLift Servers

Wenn du erwägst, deinen eigenen Spieleserver Amazon GameLift Servers zu verwalten, oder wenn du die Vorteile nutzen möchtest Amazon GameLift ServersRealtime, empfehlen wir dir, die folgenden Beispiele auszuprobieren, bevor du den Service für dein eigenes Spiel nutzt. Das Beispiel für einen benutzerdefinierten Spieleserver gibt dir Erfahrung mit dem Hosten von Spielen auf der Amazon GameLift Servers Konsole. Das Amazon GameLift Servers Realtime Beispiel zeigt dir, wie du ein Spiel für das Hosten mithilfe von Realtime Servern vorbereitest.

Beispiel für einen benutzerdefinierten Spieleserver

Dieses Beispiel zeigt den Prozess der Bereitstellung eines Beispiel-Spielerservers auf einer Amazon GameLift Servers verwalteten EC2 Flotte zum Hosten. Verwenden Sie den Beispiel-Spielclient, um eine Verbindung zu einer Live-Spielsitzung herzustellen. Sie erfahren, wie Sie Amazon GameLift Servers .tools, einschließlich der Konsole und der AWS CLI, verwenden können, um die Hosting-Leistung und -Nutzung der Flotte zu überwachen.

Das Beispiel führt Sie durch die folgenden Schritte:

- Laden Sie den Beispiel-Spielserver-Build hoch.
- Erstelle eine Flotte, um den Spielserver-Build auszuführen.
- Holen Sie sich den Beispiel-Spielclient und verwenden Sie ihn, um eine Verbindung zu einem Spieleserver herzustellen und an einer Spielsitzung teilzunehmen.
- Sieh dir die Statistiken zu Flotte und Spielsitzungen an.

Starte mehrere Spielclients und spiele das Spiel, um Hosting-Daten zu generieren. Verwende die Amazon GameLift Servers Konsole, um Hosting-Ressourcen einzusehen, Messwerte zu verfolgen und Optionen zur Skalierung der Hosting-Kapazität der Flotte zu erkunden.

Melden Sie sich zunächst bei der [Amazon GameLift ServersKonsole](#) an. Gehen Sie in der linken Navigationsleiste zu Ressourcen und probieren Sie ein Beispielspiel aus.

Aufbau einer Game-Hosting-Lösung mit Amazon GameLift Servers

Dieser Abschnitt enthält Anleitungen zum Aufbau aller Komponenten einer Amazon GameLift Servers Game-Hosting-Lösung, einschließlich bewährter Designmethoden. Ganz gleich, ob du bei Null anfängst oder ein bestehendes Spiel migrierst, hier findest du umfassende Informationen, die dir beim Aufbau, der Implementierung und Optimierung deiner Amazon GameLift Servers Game-Hosting-Infrastruktur helfen.

Komponenten der Hosting-Lösung

Ihre Hosting-Lösung muss eine Version der folgenden Komponenten enthalten. Weitere Informationen zu diesen Komponenten und ihrer Zusammenarbeit finden Sie unter [So Amazon GameLift Servers funktioniert das Hosten mit](#).

- Der Spielclient läuft auf Spielergeräten.
- Der Backend-Dienst koordiniert die Kommunikation zwischen Spieleclients und Amazon GameLift Servers.
- Die Gameserver-Software verwaltet das Multiplayer-Gameplay.
- Das Platzierungssystem ordnet Spieler den Spielservern zu.
- Spiele-Hosting-Flotten stellen Rechenressourcen an geografischen Standorten bereit.
- Das Game-Hosting-Managementsystem überwacht den Status des Spielhostings und verwaltet die Kapazität.

Themen

- [Entwicklungspläne](#)
- [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#)
- [Erstellen Sie einen Backend-Service für Amazon GameLift Servers](#)
- [Stellen Sie Hosting-Flotten bereit für Amazon GameLift Servers](#)
- [Platzierung der Spielsitzungen konfigurieren](#)
- [Passen Sie es an Ihre Game-Hosting-Lösung an](#)

Entwicklungspläne

Entwicklungs-Roadmaps bieten strukturierte step-by-step Anleitungen für die Entwicklung verschiedener Arten von Amazon GameLift Servers Game-Hosting-Lösungen. Jede Roadmap ist auf einen bestimmten Hosting-Ansatz zugeschnitten und führt Sie durch die wichtigsten Aufgaben, von der ersten Einrichtung über die Bereitstellung bis hin zum Testen.

Wählen Sie die Roadmap, die Ihren Hosting-Anforderungen und technischen Einschränkungen am besten entspricht. Sie können Ansätze auch kombinieren, indem Sie mehreren Roadmaps folgen, um eine Hybridlösung zu erstellen, die Ihren spezifischen Anforderungen entspricht.

Themen

- [Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwaltetem EC2](#)
- [Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwalteten Containern](#)
- [Entwicklungs-Roadmap für Hosting mit Anywhere Amazon GameLift Servers](#)
- [Entwicklungs-Roadmap für Hybrid-Hosting mit Amazon GameLift Servers](#)

Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwaltetem EC2

Diese Roadmap führt Sie durch die Entwicklung einer Amazon GameLift Servers verwalteten EC2-Hosting-Lösung für Ihr Multiplayer-Spiel. Amazon GameLift Servers bietet mehrere Hosting-Optionen für Spiele. Weitere Informationen zu diesen Optionen finden Sie unter [Amazon GameLift Servers Hosting-Optionen für Spiele](#)

Bei Amazon GameLift Servers verwaltetem Hosting wird dein Spieleserver auf virtuellen AWS Cloud Computerressourcen gehostet, die sich im Amazon GameLift Servers Besitz deiner Konfiguration befinden und entsprechend betrieben werden. Sie erhalten die Sicherheit, Zuverlässigkeit und globale Verfügbarkeit von Amazon Elastic Compute Cloud (Amazon EC2) -Instances, die weiter für die Verwendung mit Multiplayer-Game-Hosting optimiert sind. Amazon GameLift Servers optimiert das Hosting-Management mit Tools wie automatischen Serverbereitstellungen, Lifecycle-Handling und auto Kapazitätsskalierung.

Eine Amazon GameLift Servers verwaltete Lösung besteht aus den folgenden Komponenten:

- Ein Spieleserver-Build, der in das Server-SDK für integriert ist Amazon GameLift Servers und auf allen Flotten eingesetzt werden kann.

- Ein Spieleclient und ein Backend-Service, die in das AWS SDK integriert sind, um mit dem Amazon GameLift Servers Dienst zu interagieren und Spielsitzungen anzufordern.
- Eine Amazon GameLift Servers Warteschlange oder ein anderer Platzierungsmechanismus, um neue Spielsitzungen auf verfügbaren Spieleservern für alle Flotten zu platzieren.
- (Optional) Ein FlexMatch Matchmaker, um Mehrspieler-Matches zu erstellen und Spielsitzungen für sie einzurichten.
- Eine oder mehrere Amazon GameLift Servers verwaltete Flotten, die Amazon Elastic Compute Cloud (Amazon EC2) -Instances verwenden, die für das Hosten von Spielen im Mehrspielermodus optimiert sind.

Diese Roadmap bietet einen optimierten Weg, um Ihr Multiplayer-Spiel mit Amazon GameLift Servers verwaltetem EC2-Hosting erfolgreich zum Laufen zu bringen. Sobald Sie die erforderlichen Komponenten installiert haben, können Sie mit der Spieleentwicklung fortfahren und Ihre Hosting-Lösung individuell anpassen. Je näher die Markteinführung rückt, finden Sie hier Hilfe bei der Vorbereitung Ihrer Hosting-Lösung für den Einsatz auf Produktionsebene. [Bereite dich mit Amazon GameLift Servers Hosting auf den Start vor](#)

 Holen Sie sich einen Schnellstart mit dem Amazon GameLift Servers Plugin für Unreal Engine und Unity

Für eine schnellere Bereitstellung probieren Sie das [Amazon GameLift ServersPlugin](#) für Unreal Engine und Unity aus. Es bietet geführte UI-Workflows, mit denen du deinen Spieleserver schnell und mit minimalem Einrichtungsaufwand bereitstellen kannst, sodass du deine Spielkomponenten in Aktion ausprobieren kannst. Anschließend kannst du auf dieser Grundlage aufbauen, um eine maßgeschneiderte Hosting-Lösung für dein Spiel zu erstellen. Weitere Details finden Sie unter [Bereite dein Unreal- oder Unity-Spiel mit dem Amazon GameLift Servers Plugin vor](#).

Schritt 1: Bereite deinen Spieleserver für die Arbeit vor Amazon GameLift Servers

Füge deinem Spieleserver Funktionen hinzu, sodass er mit dem Amazon GameLift Servers Dienst kommunizieren kann, wenn er zum Hosten bereitgestellt wird.

- Hol dir das Server-SDK für Amazon GameLift Servers (Version 5.x) für dein Spielprojekt. Das Server-SDK ist in C++, C# und Go verfügbar. [Laden Sie ein Amazon GameLift Servers Server-SDK](#) herunter.

- Ändern Sie den Code Ihres Spieleservers, um Server-SDK-Funktionen hinzuzufügen. Anleitungen finden Sie unter [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#). Gehen Sie mindestens wie folgt vor:
 - Fügen Sie Code hinzu, um das Amazon GameLift Servers SDK zu initialisieren und eine WebSocket Verbindung mit dem Amazon GameLift Servers Dienst herzustellen. Verwenden Sie die Server-SDK-Aktion `InitSdk()`.
 - Fügen Sie Code hinzu, um dem Amazon GameLift Servers Dienst zu melden, wenn der Serverprozess bereit ist, Spielsitzungen zu veranstalten. Verwenden Sie die Server-SDK-Aktion `ProcessReady()`.
 - Implementieren Sie die erforderlichen `OnProcessTerminate()` Rückruffunktionen und `OnStartGameSession()`. Mit diesen Funktionen können Spieleserverprozesse eine Verbindung mit dem Amazon GameLift Servers Dienst aufrechterhalten, auf Aufforderung von Amazon GameLift Servers eine Spielsitzung einleiten und auf eine Aufforderung zum Beenden des Spieleserverprozesses reagieren.
 - Fügen Sie Code hinzu, der dem Amazon GameLift Servers Dienst meldet, wenn der Serverprozess eine Spielsitzung beendet. Verwenden Sie die Server-SDK-Aktion `ProcessEnding()`.
- Verpacke deinen Gameserver-Build. Erstelle ein Installationsskript mit deinen Build-Dateien, Abhängigkeiten und unterstützender Software. Siehe [Packen Ihrer Spielserver-Build-Dateien](#). Wir empfehlen, einen Amazon Simple Storage Service (Amazon S3) -Bucket zu verwenden, um Versionen Ihres Game-Builds zu speichern.
- Testen Sie die Integration Ihres Spieleservers. Für diese Aufgabe empfehlen wir, eine Amazon GameLift Servers Anywhere-Flotte für eine lokale Workstation einzurichten, wie unter beschrieben [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#). Installieren Sie für diesen Schritt Ihren Gameserver-Build manuell auf dem Testgerät und starten Sie einen Serverprozess. Verwenden Sie die AWS CLI, um eine neue Spielsitzung anzufordern, und stellen Sie sicher, dass der Amazon GameLift Servers Dienst Ihren Serverprozess erfolgreich zum Starten einer Spielsitzung auffordert.

Schritt 2: Bereite deinen Spielclient für die Teilnahme an gehosteten Spielsitzungen vor

Erstelle eine Möglichkeit für deinen Spielclient, die Teilnahme an einer Spielsitzung anzufordern, Verbindungsinformationen abzurufen und dann direkt eine Verbindung zu einer gehosteten Spielsitzung herzustellen. Der gängigste Ansatz besteht darin, eine Backend-Servicefunktion einzurichten, die als Vermittler zwischen deinem Spielclient und dem Dienst dient. Amazon GameLift

Servers Dieser Ansatz schützt deine Hosting-Ressourcen und gibt dir mehr Kontrolle darüber, wie Spieler an Spielsitzungen teilnehmen.

- Entwickeln Sie Backend-Servicefunktionen für das Hosting. Der Backend-Dienst kommuniziert mit dem Amazon GameLift Servers Dienst und übermittelt Verbindungsinformationen an einen Spieleclient. Diese Funktion umfasst das Starten von Spielsitzungen, das Platzieren von Spielern in Spiele und das Abrufen von Informationen zu Spielsitzungen. Anleitungen finden Sie unter [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#). Gehen Sie mindestens wie folgt vor:
 - Holen Sie sich das AWS SDK für Amazon GameLift Servers und fügen Sie es zu Ihrem Backend-Serviceprojekt hinzu. Weitere Informationen zu [Kundendiensten finden Sie unter Amazon GameLift Servers SDK-Ressourcen](#).
 - Fügen Sie Code hinzu, um einen Amazon GameLift Servers Client zu initialisieren und wichtige Einstellungen zu speichern. Siehe [Richten Sie die Amazon GameLift Servers API ein](#).
 - Fügen Sie Funktionen hinzu, um die AWS SDK-Aktion aufzurufen `CreateGameSession()` und einem Spielclient Verbindungsinformationen für die Spielsitzung bereitzustellen. Weitere Informationen [findest du unter Eine Spielsitzung auf einer bestimmten Flotte erstellen](#).

Ein Anruf `CreateGameSession()` ist ein praktischer Ausgangspunkt, um neue Spielsitzungen anzufordern. Sobald du ein System zur Platzierung von Spielsitzungen eingerichtet hast (siehe Schritt 3), ersetzt du diesen Code durch einen Anruf an `StartGameSessionPlacement()` (oder, `StartMatchmaking()` falls du es verwendest `FlexMatch`).

Anleitungen zur Gestaltung Ihres Back-End-Dienstes finden Sie unter [Erstellen Sie einen Backend-Service für Amazon GameLift Servers](#).

- Füge deinem Spielclient Funktionen hinzu, mit denen Spieler an einer gehosteten Spielsitzung teilnehmen können. Der Spielclient sendet Anfragen an deinen Backend-Service, nicht direkt an Amazon GameLift Servers. Nachdem der Backend-Dienst Verbindungsinformationen zur Spielsitzung bereitgestellt hat, stellt der Spielclient eine direkte Verbindung mit der Spielsitzung her, um das Spiel zu spielen.
- Testen Sie die Integration Ihres Spielclients. Sie können dieselbe Amazon GameLift Servers Anywhere-Flotte mit einer lokalen Workstation zum Testen verwenden.

Für eine schnelle iterative Entwicklung oder wenn Sie mit Teams aus mehreren Personen arbeiten, empfehlen wir Ihnen, eine [cloudbasierte Testumgebung](#) einzurichten. Diese Amazon GameLift Servers Toolkit-Lösung ahmt das Verhalten einer Amazon GameLift Servers verwalteten Flotte nach, ermöglicht es Ihnen jedoch, Spielserver-Builds mit minimaler Bearbeitungszeit zu aktualisieren.

Schritt 3: Richte die Platzierung der Spielsitzungen ein

Passen Sie an Amazon GameLift Servers, wie Sie Anfragen für neue Spielsitzungen bearbeiten möchten, und suchen Sie nach verfügbaren Spieleservern, auf denen sie gehostet werden können. Amazon GameLift Servers verfolgt automatisch die Verfügbarkeit aller Spieleserver auf allen Flotten. Wenn ein Spielclient eine Anfrage zur Teilnahme an einer Spielsitzung sendet, Amazon GameLift Servers sucht er anhand definierter Prioritäten wie minimale Latenz, Kosten und Verfügbarkeit nach der „bestmöglichen“ Platzierung.

- Erstelle eine Warteschlange für eine Spielsitzung, um eine neue Spielsitzung auf verfügbaren Spieleservern zu platzieren. Warteschlangen sind der wichtigste Mechanismus für die Platzierung von Spielsitzungen. Anleitungen finden Sie unter [Eine Warteschlange für Spielsitzungen erstellen](#).
- Fügen Sie Ihrer Warteschlange mindestens Ihre Anywhere-Flotten als Ziele hinzu. Bei allen anderen Einstellungen handelt es sich um optionale Anpassungen.
- Konvertieren Sie den **CreateGameSession()** Anruf in Ihrem Backend-Servicecode in **StartGameSessionPlacement()**. Weitere Informationen findest du unter [Eine Spielsitzung in einer Warteschlange mit mehreren Standorten erstellen](#).
- Erstelle einen Mechanismus, um einen Spielclient zu benachrichtigen, wenn eine Spielsitzung bereit ist, daran teilzunehmen. Während der Entwicklung kannst du den Status einer Spielsitzung abfragen, indem du einen Call to verwendest `DescribeGameSessionPlacement`. Bevor du jedoch eine Warteschlange zur Verarbeitung großer Datenmengen verwenden kannst, musst du die Ereignisbenachrichtigungen aktivieren. Siehe [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#).
- (Optional) Fügen Sie FlexMatch Matchmaking-Komponenten hinzu. Anleitungen finden Sie im [Amazon GameLift Servers FlexMatch Entwicklerhandbuch](#).

Schritt 4: Erstellen Sie verwaltete Cloud-basierte Flotten

Bis zu diesem Zeitpunkt haben Sie mit selbstverwalteten Anywhere-Flotten gearbeitet, um Ihre Spielkomponenten zu testen und zu verbessern, und Sie haben die Platzierung Ihrer Spielsitzungen verfeinert. Der letzte Teil Ihrer Lösung besteht darin, die Art der Hosting-Ressourcen einzurichten, die Sie für ein Produktionssystem benötigen. Um mit der Planung und Konfiguration für die Produktion zu beginnen, möchten Sie zur Arbeit mit einer Amazon GameLift Servers verwalteten Flotte übergehen.

- Package Sie Ihren Gameserver-Build und laden Sie ihn hoch Amazon GameLift Servers. Erstelle ein Installationsskript mit deinen Build-Dateien, Abhängigkeiten und unterstützender Software.

Siehe [Erstelle einen Gameserver-Build für Amazon GameLift Servers](#). Sie können Ihren Build entweder Amazon GameLift Servers über die Konsole oder die AWS CLI hochladen.

Bevor Sie Ihren Build hochladen, entscheiden Sie, in welchem AWS-Region Bereich Sie eine Flotte erstellen möchten. Sie müssen den Build in dieselbe Region hochladen. Weitere Informationen zur Auswahl eines Flottenstandorts finden Sie unter [Geografische Standorte](#).

- Erstellen Sie eine verwaltete EC2-Flotte. Wenn Sie eine Flotte erstellen, beginnt Amazon GameLift Servers sofort mit der Bereitstellung Ihres Gameserver-Builds für das Hosting. Du kannst viele Aspekte einer verwalteten Flotte konfigurieren. Anleitungen finden Sie unter [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#). Gehen Sie mindestens wie folgt vor:
 - Geben Sie der Flotte einen Namen und geben Sie an, welcher hochgeladene Spiel-Build eingesetzt werden soll.
 - Wählen Sie On-Demand-Instances für Ihre Flotte und wählen Sie einen Instance-Typ aus, der an Ihrem Flottenstandort verfügbar ist. Spot-Flotten sind eine wertvolle Option, erfordern jedoch zusätzliches Design und Konfiguration.
 - Erstellen Sie eine Laufzeitkonfiguration für die Flotte. Geben Sie mindestens den Startpfad für die ausführbare Datei Ihres Spieleservers an.
 - Geben Sie die Porteinstellungen an, damit eingehender Datenverkehr auf Ihre Spieleserver zugreifen kann.
- Füge die verwalteten Flotten zu deiner Warteschlange hinzu. Ersetze in deiner Warteschlange für Spielsitzungen die Anywhere-Flotten durch „deine verwalteten Flotten hinzu“.
- Testen Sie das Spiele-Hosting mit Ihren verwalteten Flotten. Zu diesem Zeitpunkt solltest du in der Lage sein, den gesamten Hosting-Zyklus zu testen. Ein Spieleclient fordert eine Spielsitzung an, ruft Verbindungsinformationen ab und stellt erfolgreich eine Verbindung zu einer Spielsitzung her.

Schritt 5: Passen Sie Ihre verwalteten Flotten an

Während du dich auf die Veröffentlichung des Spiels vorbereitest, musst du deine verwalteten Hosting-Ressourcen verfeinern. Zu den zu berücksichtigenden Entscheidungen gehören unter anderem:

- Erwägen Sie, Spot-Flotten hinzuzufügen, um Kosten zu sparen. Siehe [Senken Sie die Kosten für das Hosting von Spielen mit Spot-Flotten](#).
- Wenn dein Spieleserver andere AWS Ressourcen kommunizieren muss, richte IAM-Rollen ein, um den Zugriff zu verwalten. Siehe [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).

- Ermitteln Sie, wo Sie die Spieleserver geografisch platzieren möchten. Fügen Sie Ihren verwalteten Flotten entfernte Standorte hinzu. Siehe [Anpassungen der Hosting-Ressourcen](#).
- Optimieren Sie die Flottenleistung, indem Sie einen Instance-Typ und eine Instance-Größe auswählen und die Laufzeit so konfigurieren, dass mehrere Serverprozesse ausgeführt werden. Siehe [Optimiere die Laufzeitkonfiguration des Spieleservers auf verwalteten Amazon GameLift Servers](#).
- Experimentieren Sie mit den Optionen für die Platzierung von Spielsitzungen für verwaltete Flotten, einschließlich der Anpassung der Priorisierungseinstellungen. Siehe [Passen Sie eine Warteschlange für Spielsitzungen an](#).
- Richten Sie eine automatische Kapazitätsskalierung ein, um der erwarteten Spielernachfrage gerecht zu werden. Siehe [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#).
- Richten Sie Standby-Flotten in anderen Flotten in AWS-Regionen und ändern Sie Warteschlangen und auto Skalierung, um bei Bedarf Failover abzuwickeln.
- Richten Sie Hosting-Observability-Tools ein, einschließlich Analyse und Protokollierung. Siehe [Überwachung von Amazon GameLift Servers](#).
- Automatisieren Sie Ihre Bereitstellung mithilfe von [Infrastructure as Code \(IaC\)](#). Siehe [Amazon GameLift Servers Hosting-Ressourcen verwalten mit CloudFormation](#).

Amazon GameLift Servers unterstützt die Verwendung von AWS CloudFormation Vorlagen für beliebige bereitstellungsspezifische Konfigurationen. Sie können die auch verwenden, um Ihre AWS Cloud Development Kit (AWS CDK) Amazon GameLift Servers Ressourcen zu definieren. Weitere Informationen zu finden Sie im [AWS Cloud Development Kit \(AWS CDK\) Entwicklerhandbuch](#). AWS CDK

Um die Bereitstellung Ihrer CloudFormation Stacks zu verwalten, empfehlen wir die Verwendung von Tools und Diensten für Continuous Integration und Continuous Delivery (CI/CD), wie z. AWS CodePipeline Mit diesen Tools können Sie die Installation automatisch oder mit Genehmigung durchführen, wann immer Sie eine Binärdatei für einen Spieleserver erstellen. Mit einem CI/CD Tool oder Dienst kann die Bereitstellung von Ressourcen für eine neue Spielserverversion wie folgt aussehen:

- Erstelle und teste deine Gameserver-Binärdatei.
- Lade die Binärdatei auf hochAmazon GameLift Servers.
- Stellen Sie mit dem neuen Build neue Flotten bereit.
- Füge die neuen Flotten zu deiner Warteschlange für Spielsitzungen hinzu und entferne die Flotten mit der vorherigen Build-Version.

- Wenn die Flotten mit dem vorherigen Build keine aktiven Spielsitzungen mehr veranstalten, lösche die CloudFormation Stapel dieser Flotten.

Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwalteten Containern

Diese Roadmap führt Sie durch die Entwicklung einer Amazon GameLift Servers verwalteten Hosting-Lösung für Ihre containerisierten Spieleserver. Managed Containers ist nur eine Hosting-Lösung, die von angeboten wird. Amazon GameLift Servers Weitere Informationen zu Hosting-Optionen finden Sie unter [Amazon GameLift Servers Hosting-Optionen für Spiele](#).

Eine verwaltete Container-Lösung mit Amazon GameLift Servers den folgenden Komponenten:

- Ein Container-Image mit Ihrem Spieleserver-Build, das in das private Repository von Amazon Elastic Container Registry (Amazon ECR) hochgeladen wurde. Der Spieleserver-Build ist in das Server-SDK für Linux integriert Amazon GameLift Servers und für die Ausführung unter Linux konzipiert.
- Ein Backend-Dienst, der im Namen deiner Spieleclients mit dem Amazon GameLift Servers Dienst interagiert. Der Back-End-Dienst verwendet Funktionen in der Service-API für Amazon GameLift Servers, die Teil des SDK ist. AWS
- Eine Warteschlange für Amazon GameLift Servers Spielsitzungen oder ein anderer Platzierungsmechanismus, der Anfragen für neue Spielsitzungen verarbeitet, nach verfügbaren Spieleservern in allen Flotten sucht und einen Spieleserver auffordert, eine Spielsitzung zu starten.
- (Optional) Ein FlexMatch Matchmaker, mit dem Spiele für mehrere Spieler erstellt und Spielsitzungen für sie eingerichtet werden können.
- Eine oder mehrere Containerflotten, die Amazon Elastic Compute Cloud (Amazon EC2) -Instances verwenden und für das Hosten von Spielen im Mehrspielermodus optimiert sind.

Diese Roadmap bietet einen optimierten Weg, um Ihre containerisierten Spieleserver mit verwalteten Containern erfolgreich zum Laufen zu bringen. Amazon GameLift Servers Sobald Sie die erforderlichen Komponenten installiert haben, können Sie mit der Spieleentwicklung fortfahren und Ihre Hosting-Lösung individuell anpassen. Je näher die Markteinführung rückt, finden Sie hier Hilfe bei der Vorbereitung Ihrer Hosting-Lösung für den Einsatz auf Produktionsebene. [Bereite dich mit Amazon GameLift Servers Hosting auf den Start vor](#)

 Beschleunigen Sie das Onboarding mit diesen Tools für verwaltete Container:

- Das [Container-Starterkit](#) optimiert die Integration und die Einrichtung der Flotte. Es erweitert deinen Spieleserver um wichtige Funktionen zur Verwaltung von Spielsitzungen und verwendet vorkonfigurierte Vorlagen, um eine Containerflotte und eine automatisierte Bereitstellungspipeline für deinen Spieleserver aufzubauen. Verwende nach der Bereitstellung die Amazon GameLift Servers Konsole und die API-Tools, um die Leistung deiner Flotte zu überwachen, Spielsitzungen zu verwalten und Messwerte zu analysieren.
- Entwickler von Unreal Engine und Unity können die [Amazon GameLift ServersPlugins](#) verwenden, um ihren Spieleserver zu integrieren und aus der Entwicklungsumgebung Ihrer Game Engine heraus eine Containerflotte aufzubauen. Die geführten Workflows des Plugins helfen dir dabei, eine schnelle, einfache Lösung mit cloudbasiertem Hosting mithilfe verwalteter Container zu erstellen. Du kannst auf dieser Grundlage aufbauen, um eine maßgeschneiderte Hosting-Lösung für dein Spiel zu erstellen.

Schritt 1: Bereite deinen Spieleserver für die Arbeit vor Amazon GameLift Servers

Füge deinem Spieleserver Funktionen hinzu, damit er mit dem Amazon GameLift Servers Dienst kommunizieren kann, wenn er zum Hosten bereitgestellt wird.

- Besorgen Sie sich das Server-SDK für Amazon GameLift Servers (Version 5.2 oder höher) für Ihr Spielprojekt. Das Server-SDK ist in C++, C# und Go verfügbar. [Laden Sie das Server-SDK für](#) herunterAmazon GameLift Servers. Das Server-SDK ist in C++, C# und Go verfügbar.
- Ändern Sie den Code Ihres Spieleservers, um Server-SDK-Funktionen hinzuzufügen. Anleitungen finden Sie unter [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#). Gehen Sie mindestens wie folgt vor:
 - Fügen Sie Code hinzu, um das Amazon GameLift Servers SDK zu initialisieren und eine WebSocket Verbindung mit dem Amazon GameLift Servers Dienst herzustellen. Verwenden Sie die Server-SDK-Aktion `InitSdk()`.
 - Fügen Sie Code hinzu, um dem Amazon GameLift Servers Dienst zu melden, wenn der Serverprozess bereit ist, Spielsitzungen zu veranstalten. Verwenden Sie die Server-SDK-Aktion `ProcessReady()`.
 - Implementieren Sie die erforderlichen `OnStartGameSession()` Rückruffunktionen und `OnProcessTerminate()`. Mit diesen Funktionen können Spieleserverprozesse eine Verbindung mit dem Amazon GameLift Servers Dienst aufrechterhalten, auf Aufforderung eine

Spielsitzung einleiten und auf eine Anfrage zum Beenden des Spielserverprozesses antworten.

Amazon GameLift Servers

- Fügen Sie Code hinzu, um dem Amazon GameLift Servers Dienst zu melden, wenn der Serverprozess eine Spielsitzung beendet. Verwenden Sie die Server-SDK-Aktion `ProcessEnding()`.
- Verpacke deinen Gameserver-Build. Baue deinen Gameserver so, dass er unter Linux läuft. Bereite den Build und andere Dateien vor, die für den Betrieb des Spieleservers benötigt werden. Wenn du unter Windows entwickelst, beinhaltet dieser Schritt möglicherweise die Einrichtung eines separaten Linux-Workspace oder die Verwendung eines Tools wie Windows Subsystem for Linux (WSL). Du benötigst eine Linux-Umgebung, um deinen Gameserver-Build zu testen und deine Container-Images zu erstellen und zu testen.
- Testen Sie Ihre Gameserver-Integration. Vergewissere dich, dass dein integrierter Spieleserver eine Verbindung zum Amazon GameLift Servers Dienst herstellen und auf Eingabeaufforderungen reagieren kann. Wir empfehlen, eine einfache Amazon GameLift Servers Anywhere-Flotte mit einer lokalen Workstation als Testhost einzurichten, wie unter beschrieben [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#). Installieren Sie Ihren Gameserver-Build auf dem Testhost und starten Sie einen Serverprozess. Verwenden Sie die AWS CLI, um eine neue Spielsitzung anzufordern, und stellen Sie sicher, dass der Amazon GameLift Servers Dienst Ihren Serverprozess erfolgreich zum Starten einer Spielsitzung auffordert.

Schritt 2: Erstellen Sie ein Container-Image für den Spieleserver

Nachdem du deinen Spieleserver erfolgreich integriert hast, erstelle ein Container-Image mit der ausführbaren Datei deines Spieleservers. Speichern Sie es zur Verwendung in einem Amazon Elastic Container Registry (Amazon ECR) -Repository. Amazon GameLift Servers Detaillierte Anweisungen finden Sie unter [Erstellen Sie ein Container-Image für Amazon GameLift Servers](#).

- Holen Sie sich die Dockerfile-Vorlage für einen Spieleserver-Container (bereitgestellt von). Amazon GameLift Servers Ändern Sie die Datei für die Build-Dateien Ihres Spieleservers.
- Erstellen Sie ein Container-Image für den Spieleserver. Wenn Sie in einer Linux-Umgebung arbeiten, verwenden Sie das Docker-Tool, um Ihr Image zu erstellen.
- Übertragen Sie Ihr Container-Image auf Amazon ECR. Erstellen Sie ein öffentliches oder privates Repository in Amazon ECR, indem Sie dasselbe verwenden AWS-Konto und AWS-Region wo Sie Ihre Containerflotte einsetzen möchten. Pushen Sie Ihr Container-Image dorthin.
- Testen Sie Ihre Container-Images mit Ihrer Anywhere-Flotte (optional). Möglicherweise möchten Sie Ihre Container-Images lokal testen, bevor Sie sie in einer in der Cloud gehosteten

Containerflotte bereitstellen. Sie können Ihre bestehende Amazon GameLift Servers Anywhere-Flotte mit einer lokalen Workstation zum Testen verwenden. Installieren und starten Sie den Spielserver-Container und stellen Sie sicher, dass: (1) der Amazon GameLift Servers Dienst Ihren Serverprozess erfolgreich auffordert, eine Spielsitzung zu starten, und (2) ein Spielclient eine Verbindung zur Spielsitzung herstellen kann.

Schritt 3: Erstellen Sie eine cloudbasierte Containerflotte

Bis zu diesem Zeitpunkt hast du mit einer selbstverwalteten Anywhere-Flotte gearbeitet, um deine Spielkomponenten zu testen und zu iterieren. Wenn Sie über einen funktionierenden Gameserver-Build verfügen, der integriert ist Amazon GameLift Servers, können Sie mit der Einrichtung der cloudbasierten Amazon GameLift Servers verwalteten Containerflotte beginnen, die Ressourcen hostet, die Sie für eine Produktionsumgebung benötigen.

- Erstellen Sie Definitionen für Containergruppen. Container-Gruppendefinitionen beschreiben die Container-Architektur für eine Flotte und identifizieren, welche Container-Images bereitgestellt werden sollen. Siehe [Erstellen Sie eine Containergruppendefinition für eine Amazon GameLift Servers Containerflotte](#). Erstellen Sie Ihre Container-Gruppendefinition dort, AWS-Region wo die Container-Images gespeichert sind. Weitere Informationen zur Auswahl eines Flottenstandorts finden Sie unter [Geografische Standorte](#). Gehen Sie mindestens wie folgt vor:
 - Erstellen Sie eine Spielserver-Container-Gruppendefinition.
 - Fügen Sie eine Container-Definition mit einem Container-Image zu Ihrem Gameserver-Build hinzu.
 - Konfigurieren Sie einen Portbereich für die Gameserver-Prozesse des Containers.
- Erstellen Sie eine verwaltete Containerflotte. Wenn du eine Flotte erstellst, beginnt Amazon GameLift Servers sofort mit der Bereitstellung deines Gameserver-Builds für das Hosting. Du kannst viele Aspekte einer verwalteten Flotte konfigurieren. Anleitungen finden Sie unter [Erstellen Sie eine Amazon GameLift Servers verwaltete Containerflotte](#). Gehen Sie mindestens wie folgt vor:
 - Richten Sie eine AWS Identity and Access Management (IAM) -Servicerolle für die Containerflotte ein. Siehe [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#).
 - Geben Sie die Definition der Spielserver-Containergruppe an, die auf Flotteninstanzen bereitgestellt werden soll.
 - Verwenden Sie Standardwerte, sofern verfügbar, für alle anderen Parameter. Amazon GameLift Servers berechnet einige Parameter für eine optimale Konfiguration.

- Fügen Sie die Containerflotten zu Ihrer Warteschlange hinzu. Ersetze in deiner Warteschlange für Spielsitzungen die Anywhere-Testflotte durch deine verwaltete Containerflotte.
- Testen Sie das Game-Hosting mit Ihren Containerflotten. Zu diesem Zeitpunkt sollten Sie in der Lage sein, die gesamte Lösung zu testen. Starte einen Spielclient und fordere über den Backend-Dienst eine Spielsitzung an. Holen Sie sich Verbindungsinformationen und stellen Sie eine Verbindung zu einer Spielsitzung auf der Containerflotte her.
- Iteriere deine Flotteneinsätze. Sie können Containergruppenseite und Flottenkonfigurationen aktualisieren und anschließend Updates für bestehende Flotten bereitstellen.

Schritt 4: Bereite deinen Spielclient für die Teilnahme an gehosteten Spielsitzungen vor

Erstelle eine Möglichkeit für deinen Spielclient, die Teilnahme an einer Spielsitzung anzufordern, Verbindungsinformationen abzurufen und dann direkt eine Verbindung zu einer gehosteten Spielsitzung herzustellen. Der gängigste Ansatz besteht darin, eine Backend-Servicefunktion einzurichten, die als Vermittler zwischen deinem Spielclient und dem Dienst dient. Amazon GameLift Servers Dieser Ansatz schützt deine Hosting-Ressourcen und gibt dir mehr Kontrolle darüber, wie Spieler an Spielsitzungen teilnehmen.

- Entwickeln Sie Backend-Servicefunktionen für das Hosting. Der Backend-Dienst kommuniziert mit dem Amazon GameLift Servers Dienst und übermittelt Verbindungsinformationen an einen Spielclient. Diese Funktion umfasst das Starten von Spielsitzungen, das Platzieren von Spielern in Spiele und das Abrufen von Informationen zu Spielsitzungen. Anleitungen finden Sie unter [Bereite ein Spiel für das Hosting vor mit Amazon GameLift Servers](#). Gehen Sie mindestens wie folgt vor:
 - Holen Sie sich das AWS SDK für Amazon GameLift Servers und fügen Sie es Ihrem Backend-Serviceprojekt hinzu. Weitere Informationen zu [Kundendiensten finden Sie unter Amazon GameLift Servers SDK-Ressourcen](#).
 - Fügen Sie Code hinzu, um einen Amazon GameLift Servers Client zu initialisieren und wichtige Einstellungen zu speichern. Siehe [Richten Sie die Amazon GameLift Servers API ein](#).
 - Fügen Sie Funktionen hinzu, um die AWS SDK-Aktion aufzurufen `CreateGameSession()` und einem Spielclient Verbindungsinformationen für die Spielsitzung bereitzustellen. Weitere Informationen [findest du unter Eine Spielsitzung auf einer bestimmten Flotte erstellen](#).

Ein Aufruf `CreateGameSession()` ist ein praktischer Ausgangspunkt, um neue Spielsitzungen anzufordern. Sobald du ein System zur Platzierung von Spielsitzungen eingerichtet hast (siehe Schritt 3), ersetzt du diesen Code durch einen Aufruf an `StartGameSessionPlacement()` (oder, `StartMatchmaking()` falls du es verwendest `FlexMatch`).

Anleitungen zur Gestaltung Ihres Back-End-Dienstes finden Sie unter [Erstellen Sie einen Backend-Service für Amazon GameLift Servers](#).

- Füge deinem Spielclient Funktionen hinzu, mit denen Spieler an einer gehosteten Spielsitzung teilnehmen können. Der Spielclient sendet Anfragen an deinen Backend-Service, nicht direkt an Amazon GameLift Servers. Nachdem der Backend-Dienst Verbindungsinformationen zur Spielsitzung bereitgestellt hat, stellt der Spielclient eine direkte Verbindung mit der Spielsitzung her, um das Spiel zu spielen.
- Testen Sie die Integration Ihres Spielclients. Sie können Ihre bestehende Amazon GameLift Servers Anywhere-Flotte mit einer lokalen Workstation zum Testen verwenden. Verwenden Sie den neuen Backend-Dienst, um eine neue Spielsitzung anzufordern, und stellen Sie sicher, dass: (1) der Amazon GameLift Servers Dienst Ihren Serverprozess erfolgreich auffordert, eine Spielsitzung zu starten, und (2) ein Spielclient eine Verbindung zur Spielsitzung herstellen kann.

Schritt 5: Richte die Platzierung der Spielsitzung ein

Passen Sie an Amazon GameLift Servers, wie Sie Anfragen für neue Spielsitzungen bearbeiten möchten, und suchen Sie nach verfügbaren Spieleservern, auf denen sie gehostet werden können. Amazon GameLift Servers verfolgt automatisch die Verfügbarkeit aller Spieleserver auf allen Flotten. Wenn ein Spielclient eine Anfrage zur Teilnahme an einer Spielsitzung sendet, Amazon GameLift Servers sucht er anhand definierter Prioritäten wie minimale Latenz, Kosten und Verfügbarkeit nach der „bestmöglichen“ Platzierung.

- Erstelle eine Warteschlange für eine Spielsitzung, um eine neue Spielsitzung auf verfügbaren Spieleservern zu platzieren. Warteschlangen sind der wichtigste Mechanismus für die Platzierung von Spielsitzungen. Anleitungen finden Sie unter [Eine Warteschlange für Spielsitzungen erstellen](#).
- Fügen Sie Ihrer Warteschlange mindestens Ihre Anywhere-Flotten als Ziele hinzu. Alle anderen Einstellungen sind optionale Anpassungen.
- Konvertieren Sie den **CreateGameSession()** Anruf in Ihrem Backend-Servicecode in **StartGameSessionPlacement()**. Weitere Informationen findest du unter [Eine Spielsitzung in einer Warteschlange mit mehreren Standorten erstellen](#).
- Erstelle einen Mechanismus, um einen Spielclient zu benachrichtigen, wenn eine Spielsitzung bereit ist, daran teilzunehmen. Während der Entwicklung kannst du den Status einer Spielsitzung abfragen, indem du einen Call to verwendest **DescribeGameSessionPlacement**. Bevor du jedoch eine Warteschlange zur Verarbeitung großer Datenmengen verwenden kannst, musst du

die Ereignisbenachrichtigungen aktivieren. Siehe [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#).

- Fügen Sie FlexMatch Matchmaking hinzu (optional). Erstelle einen Regelsatz für die Spielerzuweisung und erstelle eine Matchmaking-Konfiguration, die mit deiner Warteschlange für Spielsitzungen funktioniert. [Anleitungen zur Einrichtung eines Matchmaking-Systems findest du im Entwicklerhandbuch. Amazon GameLift ServersFlexMatch](#)
- Testen Sie das Platzierungssystem. Sie können Ihre bestehende Amazon GameLift Servers Anywhere-Flotte mit einer lokalen Workstation zum Testen verwenden. Verwenden Sie den Back-End-Dienst, um eine neue Spielsitzung anzufordern, und stellen Sie sicher, dass der Amazon GameLift Servers Dienst Ihren Serverprozess erfolgreich zum Starten einer Spielsitzung auffordert.

Schritt 6: Passen Sie Ihre verwaltete Container-Lösung an

Während du dich auf die Veröffentlichung des Spiels vorbereitest, musst du deine verwalteten Hosting-Ressourcen optimieren. Zu den zu berücksichtigenden Entscheidungen gehören unter anderem:

- Optimieren Sie die Konfiguration Ihrer Containerflotte. Siehe [Passen Sie eine Amazon GameLift Servers Containerflotte an](#).
- Erwägen Sie, Spot-Flotten hinzuzufügen, um Kosten zu sparen. Siehe [Senken Sie die Kosten für das Hosting von Spielen mit Spot-Flotten](#).
- Wenn dein Spieleserver andere AWS Ressourcen kommunizieren muss, richte IAM-Rollen ein, um den Zugriff zu verwalten. Siehe [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).
- Ermitteln Sie, wo Sie die Spieleserver geografisch platzieren möchten. Fügen Sie Ihren verwalteten Flotten entfernte Standorte hinzu. Siehe [Anpassungen der Hosting-Ressourcen](#).
- Experimentieren Sie mit den Optionen für die Platzierung von Spielsitzungen für verwaltete Flotten, einschließlich der Anpassung der Priorisierungseinstellungen. Siehe [Passen Sie eine Warteschlange für Spielsitzungen an](#).
- Richten Sie eine automatische Kapazitätsskalierung ein, um der erwarteten Spielernachfrage gerecht zu werden. Siehe [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#).
- Erstellen Sie Flotten in anderen Flotten AWS-Regionen und ändern Sie Warteschlangen und auto Skalierung, um Failover nach Bedarf abzuwickeln.
- Richten Sie Hosting-Observability-Tools ein, einschließlich Analyse und Protokollierung. Siehe [Überwachung von Amazon GameLift Servers](#).

- Automatisieren Sie Ihre Flottenbereitstellungen mithilfe von [Infrastructure as Code \(IaC\)](#). Siehe [Amazon GameLift Servers Hosting-Ressourcen verwalten mit CloudFormation](#).

Amazon GameLift Servers unterstützt die Verwendung von AWS CloudFormation Vorlagen für alle einsatzspezifischen Konfigurationen. Sie können die auch verwenden, um Ihre AWS Cloud Development Kit (AWS CDK) Amazon GameLift Servers Ressourcen zu definieren. Weitere Informationen zu finden Sie im [AWS Cloud Development Kit \(AWS CDK\) Entwicklerhandbuch](#).
AWS CDK

Um die Bereitstellung Ihrer CloudFormation Stacks zu verwalten, empfehlen wir die Verwendung von Tools und Diensten für Continuous Integration und Continuous Delivery (CI/CD) wie. AWS CodePipeline Mit diesen Tools können Sie die Installation automatisch oder mit Genehmigung durchführen, wann immer Sie eine Binärdatei für einen Spielservers erstellen. Mit einem CI/CD Tool oder Dienst kann die Bereitstellung von Ressourcen für eine neue Spielserversversion wie folgt aussehen:

- Erstelle und teste deine Gameserver-Binärdatei.
- Lade die Binärdatei auf hochAmazon GameLift Servers.
- Stellen Sie mit dem neuen Build neue Flotten bereit.
- Füge die neuen Flotten zu deiner Warteschlange für Spielsitzungen hinzu und entferne die Flotten mit der vorherigen Build-Version.
- Wenn die Flotten mit dem vorherigen Build keine aktiven Spielsitzungen mehr veranstalten, lösche die CloudFormation Stapel dieser Flotten.

Entwicklungs-Roadmap für Hosting mit Anywhere Amazon GameLift Servers

Diese Roadmap führt Sie durch die Entwicklung einer Hosting-Lösung für Ihr Multiplayer-Spiel, die Sie mit Ihren eigenen Ressourcen (lokale Hardware oder virtuelle Maschinen) verwenden können. Amazon GameLift Servers bietet mehrere Hosting-Optionen für Spiele. Weitere Informationen zu diesen Optionen finden Sie unter. [Amazon GameLift Servers Hosting-Optionen für Spiele](#)

Beim Amazon GameLift Servers Anywhere-Hosting wird Ihr Spielservers auf Computerressourcen gehostet, die Sie bereitstellen und verwalten. Sie können eine Anywhere-Flotte mit den Konfigurationen erstellen, die Sie benötigen, und die sich geografisch unabhängig vom Standort Ihrer Spieler befindet. Amazon GameLift Servers bietet die folgenden Funktionen für eine Anywhere-Flotte:

- Übernimmt für Sie die Platzierung von Spielsitzungen auf der Grundlage Ihrer Konfiguration, einschließlich:
 - Verfolgt die Verfügbarkeit von Spielservern in Ihren Anywhere-Flotten.
 - Verarbeitung von Spielanfragen von Ihrem Spiele-Client-Dienst und Abgleich von Spielanfragen mit verfügbaren Servern.
 - Spieleserver auf Anywhere-Flotten werden aufgefordert, Spielsitzungen zu starten.
 - Übermittlung der Verbindungsdetails an die Spieleclients.
- Sammelt Leistungskennzahlen für den Prozess der Sitzungsplanung und Nutzungsmetriken für Spielsitzungen und Spieler.
- Unterstützt den vollen Funktionsumfang der FlexMatch Spielersuche, sodass Sie einen Matchmaker erstellen und ihn in das Platzierungssystem für Spielsitzungen integrieren können.
- Bietet dem Amazon GameLift Servers Agenten die Möglichkeit, wichtige Aufgaben der Hostverwaltung auf einer Anywhere-Flotte zu erledigen.
- Unterstützt die Kombination mit Amazon GameLift Servers verwalteten Flotten für eine flexible Hybridlösung.

Eine Amazon GameLift Servers Anywhere-Lösung besteht aus den folgenden Komponenten:

- Ein Spieleserver-Build, der in das Server-SDK für integriert ist Amazon GameLift Servers und auf allen Flotten eingesetzt werden kann.
- Ein Spielclient und ein Backend-Service, die in das AWS SDK integriert sind, um mit dem Amazon GameLift Servers Dienst zu interagieren und Spielsitzungen anzufordern.
- Eine Amazon GameLift Servers Warteschlange oder ein anderer Platzierungsmechanismus, um neue Spielsitzungen auf verfügbaren Spieleservern für alle Flotten zu platzieren.
- (Optional) Ein FlexMatch Matchmaker, um Spiele für mehrere Spieler zu erstellen und Spielsitzungen für sie einzurichten.
- Eine oder mehrere Amazon GameLift Servers Anywhere-Flotten mit Ihren lokalen oder anderen Hosting-Ressourcen, verwaltet mit Ihren vorhandenen Tools für Konfigurationsmanagement und Bereitstellung. (Sie können optional die verwenden.) AWS Systems Manager

Diese Roadmap bietet einen optimierten Weg, um Ihr Multiplayer-Spiel mit Amazon GameLift Servers Anywhere-Hosting erfolgreich zum Laufen zu bringen. Sobald Sie die erforderlichen Komponenten installiert haben, können Sie mit der Spieleentwicklung fortfahren und Ihre Hosting-Lösung individuell anpassen. Je näher die Markteinführung rückt, finden Sie hier Hilfe bei der Vorbereitung Ihrer

Hosting-Lösung für den Einsatz auf Produktionsebene. [Bereite dich mit Amazon GameLift Servers Hosting auf den Start vor](#)

-  Holen Sie sich einen Schnellstart mit dem Amazon GameLift Servers Plugin für Unreal Engine und Unity

Für eine schnellere Bereitstellung probieren Sie das [Amazon GameLift ServersPlugin](#) für Unreal Engine und Unity aus. Es bietet geführte UI-Workflows, mit denen du deinen Spieleserver mit minimalem Einrichtungsaufwand schnell bereitstellen kannst, sodass du deine Spielkomponenten in Aktion ausprobieren kannst. Anschließend kannst du auf dieser Grundlage aufbauen, um eine maßgeschneiderte Hosting-Lösung für dein Spiel zu erstellen. Weitere Details finden Sie unter [Bereite dein Unreal- oder Unity-Spiel mit dem Amazon GameLift Servers Plugin vor](#).

Schritt 1: Bereite deinen Spieleserver für die Arbeit vor Amazon GameLift Servers

Füge deinem Spieleserver Funktionen hinzu, damit er mit dem Amazon GameLift Servers Dienst kommunizieren kann, wenn er zum Hosten bereitgestellt wird.

- Hol dir das Server-SDK für Amazon GameLift Servers (Version 5.x) für dein Spielprojekt. Das Server-SDK ist in C++, C# und Go verfügbar. [Laden Sie ein Amazon GameLift Servers Server-SDK](#) herunter.
- Ändern Sie den Code Ihres Spieleservers, um Server-SDK-Funktionen hinzuzufügen. Anleitungen finden Sie unter [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#). Gehen Sie mindestens wie folgt vor:
 - Fügen Sie Code hinzu, um das Amazon GameLift Servers SDK zu initialisieren und eine WebSocket Verbindung mit dem Amazon GameLift Servers Dienst herzustellen. Verwenden Sie die Server-SDK-Aktion `InitSdk()` und fügen Sie Serverparameter hinzu, die für eine Anywhere-Flotte erforderlich sind.
 - Fügen Sie Code hinzu, um dem Amazon GameLift Servers Dienst zu melden, wenn der Serverprozess bereit ist, Spielsitzungen abzuhalten. Verwenden Sie die Server-SDK-Aktion `ProcessReady()`.
 - Implementieren Sie die erforderlichen `OnProcessTerminate()` Rückruffunktionen und `OnStartGameSession()`. Mit diesen Funktionen können Spieleserverprozesse eine Verbindung mit dem Amazon GameLift Servers Dienst aufrechterhalten, auf Aufforderung von

Amazon GameLift Servers eine Spielsitzung einleiten und auf eine Aufforderung zum Beenden des Spielserverprozesses reagieren.

- Fügen Sie Code hinzu, der dem Amazon GameLift Servers Dienst meldet, wenn der Serverprozess eine Spielsitzung beendet. Verwenden Sie die Server-SDK-Aktion `ProcessEnding()`.
- Verpacke deinen Gameserver-Build. Erstelle ein Installationsskript mit deinen Build-Dateien, Abhängigkeiten und unterstützender Software. Siehe [Packen Ihrer Spielserver-Build-Dateien](#). Wir empfehlen, einen Amazon Simple Storage Service (Amazon S3) -Bucket zu verwenden, um Versionen Ihres Game-Builds zu speichern.
- Testen Sie Ihre Gameserver-Integration. Für diese Aufgabe empfehlen wir, eine Amazon GameLift Servers Anywhere-Flotte für eine lokale Workstation einzurichten, wie unter beschrieben [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#). Installieren Sie für diesen Schritt Ihren Gameserver-Build manuell auf dem Testgerät und starten Sie einen Serverprozess. Verwenden Sie die AWS CLI, um eine neue Spielsitzung anzufordern, und stellen Sie sicher, dass der Amazon GameLift Servers Dienst Ihren Serverprozess erfolgreich zum Starten einer Spielsitzung auffordert.

Schritt 2: Bereite deinen Spielclient für die Teilnahme an gehosteten Spielsitzungen vor

Erstelle eine Möglichkeit für deinen Spielclient, die Teilnahme an einer Spielsitzung anzufordern, Verbindungsinformationen abzurufen und dann direkt eine Verbindung zu einer gehosteten Spielsitzung herzustellen. Der gängigste Ansatz besteht darin, eine Backend-Servicefunktion einzurichten, die als Vermittler zwischen deinem Spielclient und dem Dienst dient. Amazon GameLift Servers Dieser Ansatz schützt deine Hosting-Ressourcen und gibt dir mehr Kontrolle darüber, wie Spieler an Spielsitzungen teilnehmen.

- Entwickeln Sie Backend-Servicefunktionen für das Hosting. Der Backend-Dienst kommuniziert mit dem Amazon GameLift Servers Dienst und übermittelt Verbindungsinformationen an einen Spielclient. Diese Funktion umfasst das Starten von Spielsitzungen, das Platzieren von Spielern in Spiele und das Abrufen von Informationen zu Spielsitzungen. Anleitungen finden Sie unter [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#). Gehen Sie mindestens wie folgt vor:
 - Holen Sie sich das AWS SDK für Amazon GameLift Servers und fügen Sie es zu Ihrem Backend-Serviceprojekt hinzu. Siehe [Amazon GameLift Servers SDK-Ressourcen für Kundendienste](#).
 - Fügen Sie Code hinzu, um einen Amazon GameLift Servers Client zu initialisieren und wichtige Einstellungen zu speichern. Siehe [Richten Sie die Amazon GameLift Servers API ein](#).

- Fügen Sie Funktionen hinzu, um die AWS SDK-Aktion aufzurufen `CreateGameSession()` und einem Spielclient Verbindungsinformationen für die Spielsitzung bereitzustellen. Weitere Informationen [findest du unter Eine Spielsitzung auf einer bestimmten Flotte erstellen](#).

Ein Anruf `CreateGameSession()` ist ein praktischer Ausgangspunkt, um neue Spielsitzungen anzufordern. Sobald du ein System zur Platzierung von Spielsitzungen eingerichtet hast (siehe Schritt 3), ersetzt du diesen Code durch einen Anruf an `StartGameSessionPlacement()` (oder, `StartMatchmaking()` falls du es verwendest `FlexMatch`).

Anleitungen zur Gestaltung Ihres Back-End-Dienstes finden Sie unter [Erstellen Sie einen Backend-Service für Amazon GameLift Servers](#).

- Füge deinem Spielclient Funktionen hinzu, mit denen Spieler an einer gehosteten Spielsitzung teilnehmen können. Der Spielclient sendet Anfragen an deinen Backend-Service, nicht direkt an Amazon GameLift Servers. Nachdem der Backend-Dienst Verbindungsinformationen zur Spielsitzung bereitgestellt hat, stellt der Spielclient eine direkte Verbindung mit der Spielsitzung her, um das Spiel zu spielen.
- Testen Sie die Integration Ihres Spielclients. Sie können dieselbe Amazon GameLift Servers Anywhere-Flotte mit lokalen Workstations zum Testen verwenden.

Schritt 3: Richten Sie die Platzierung der Spielsitzungen ein

Passen Sie an Amazon GameLift Servers, wie Sie Anfragen für neue Spielsitzungen bearbeiten möchten, und suchen Sie nach verfügbaren Spieleservern, auf denen sie gehostet werden können. Amazon GameLift Servers verfolgt automatisch die Verfügbarkeit aller Spieleserver auf allen Flotten. Wenn ein Spielclient eine Anfrage zur Teilnahme an einer Spielsitzung sendet, Amazon GameLift Servers sucht er anhand definierter Prioritäten wie minimale Latenz, Kosten und Verfügbarkeit nach der „bestmöglichen“ Platzierung.

- Erstelle eine Warteschlange für eine Spielsitzung, um eine neue Spielsitzung auf verfügbaren Spieleservern zu platzieren. Warteschlangen sind der wichtigste Mechanismus für die Platzierung von Spielsitzungen. Anleitungen finden Sie unter [Eine Warteschlange für Spielsitzungen erstellen](#).
- Fügen Sie Ihrer Warteschlange mindestens Ihre Anywhere-Flotten als Ziele hinzu. Alle anderen Einstellungen sind optional.
- Konvertieren Sie den **`CreateGameSession()`** Anruf in Ihrem Backend-Servicecode in **`StartGameSessionPlacement()`** Weitere Informationen findest du unter [Eine Spielsitzung in einer Warteschlange mit mehreren Standorten erstellen](#).

- Erstelle einen Mechanismus, um einen Spielclient zu benachrichtigen, wenn eine Spielsitzung bereit ist, daran teilzunehmen. Während der Entwicklung kannst du den Status einer Spielsitzung abfragen, indem du einen Call to verwendest `DescribeGameSessionPlacement`. Bevor du jedoch eine Warteschlange zur Verarbeitung großer Datenmengen verwenden kannst, musst du die Ereignisbenachrichtigungen aktivieren. Siehe [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#).
- (Optional) Fügen Sie FlexMatch Matchmaking-Komponenten hinzu. Anleitungen finden Sie im [Amazon GameLift Servers FlexMatch Entwicklerhandbuch](#).

Schritt 4: Richten Sie mit dem Amazon GameLift Servers Agenten eine Anywhere-Flotte ein

Bis zu diesem Zeitpunkt hast du mit lokalen Geräten (registriert als Anywhere Fleet Computes) gearbeitet, um deine Spielkomponenten zu testen und zu verbessern. Der nächste Schritt besteht darin, die Art von Flotte einzurichten, die Sie für ein Produktionssystem benötigen. Verwenden Sie für diese Ressourcen den Amazon GameLift Servers Agenten, um einige wichtige Aufgaben zur Serververwaltung am Computer zu verwalten. Weitere Details finden Sie unter [Arbeite mit dem Amazon GameLift Servers Agenten](#).

- Hol dir den Amazon GameLift Servers Agenten und füge ihn deinem Gameserver-Installationspaket hinzu. Holen Sie sich den Agent-Quellcode, der im [Github-Repository für Amazon GameLift Servers Agenten verfügbar ist, und erstellen Sie ihn](#). Platzieren Sie die resultierende ausführbare JAR-Datei in demselben Verzeichnis wie die ausführbare Datei für den Spiel-Build.
- Ändern Sie Ihr Startskript für den Agenten nach Bedarf. Stellen Sie sicher, dass die ausführbare Datei des Agenten gestartet wird, sobald ein Computer ausgeführt wird. Hilfe zur Installation und Ausführung des Agenten auf Ihren Hosting-Computern finden Sie in der Readme-Datei im Agent-Repository. Ihr Startbefehl sollte Optionen enthalten, mit denen Sie mindestens die Anywhere-Flotten-ID sowie einen benutzerdefinierten Standort und AWS-Region einen Computernamen angeben können.

Der Agent erledigt automatisch die folgenden Aufgaben für Sie. Wenn Sie diese Aufgaben also mit Skripten erledigt haben, können Sie sie entfernen:

- Ruft `RegisterCompute()` auf, um die Rechenleistung einer Anywhere-Flotte hinzuzufügen.
- Aufrufe `GetComputeAuthToken()` zur Authentifizierung von Spieleservern, wenn sie sich mit dem Amazon GameLift Servers Dienst verbinden. Der Agent verwaltet das Abrufen und Aktualisieren des Authentifizierungstokens, das von allen Spieleserverprozessen verwendet werden kann, die auf dem Computer ausgeführt werden.

- Startet neue Serverprozesse auf dem Computer auf der Grundlage einer Reihe von Laufzeitanweisungen.
- Erstellen Sie eine Laufzeitkonfiguration für Computer in Ihrer Anywhere-Flotte. Sie können die Amazon GameLift Servers Konsole oder die AWS CLI verwenden, um Laufzeitanweisungen für die Flotte zu erstellen oder zu ändern. Der Agent führt diese Anweisungen aus und fordert regelmäßig Updates vom Amazon GameLift Servers Service an.
- Richten Sie Ihre Warteschlange für Spielsitzungen nach Bedarf ein oder ändern Sie sie. Erstellen Sie eine neue Warteschlange (oder aktualisieren Sie eine bestehende), um die Anywhere-Flotten zu verwenden, die zusammen mit dem Amazon GameLift Servers Agenten eingesetzt werden.
- Testen Sie die Agent-Integration mit Ihren Anywhere-Flotten. Stellen Sie sicher, dass der Agent die Serverprozesse auf der Grundlage der Laufzeitkonfiguration ordnungsgemäß startet.

Schritt 5: Passen Sie Ihre Anywhere-Flotten an

Während du dich auf die Veröffentlichung des Spiels vorbereitest, musst du deine verwalteten Hosting-Ressourcen optimieren. Zu den zu berücksichtigenden Entscheidungen gehören unter anderem:

- Automatisieren Sie den Prozess des Startens und Herunterfahrens von Computern nach Bedarf, einschließlich der Installation und Ausführung von Gameserver-Software. Das Wiederverwenden von Computern ist nützlich, um sicherzustellen, dass sie regelmäßig aktualisiert werden. Durch das Herunterfahren von Computern können Kosten eingespart werden, wenn sie nicht benötigt werden.
- Wenn dein Spieleserver andere AWS Ressourcen kommunizieren muss, richte IAM-Rollen ein, um den Zugriff zu verwalten. Siehe [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).
- Ermitteln Sie, wo Sie die Spieleserver geografisch platzieren möchten. Fügen Sie Ihren verwalteten Flotten entfernte Standorte hinzu. Siehe [Anpassungen der Hosting-Ressourcen](#).
- Optimieren Sie die Flottenleistung, indem Sie Rechenressourcenkonfigurationen auswählen und anschließend die Laufzeitanweisungen so konfigurieren, dass eine optimale Anzahl von Serverprozessen pro Computer ausgeführt wird.
- Experimentieren Sie mit den Optionen für die Platzierung von Spielsitzungen für verwaltete Flotten, einschließlich der Anpassung der Priorisierungseinstellungen. Siehe [Passen Sie eine Warteschlange für Spielsitzungen an](#).
- Entwickeln Sie Mechanismen für die manuelle oder automatische Kapazitätsskalierung, um der erwarteten Spielernachfrage gerecht zu werden. Überlegen Sie, welche Faktoren das System

veranlassen sollten, die Anzahl der Computer, die für die Durchführung von Spielsitzungen zur Verfügung stehen, zu erhöhen oder zu verringern.

- Entwerfen und implementieren Sie bei Bedarf einen Failover auf andere Ressourcen.
- Richten Sie Hosting-Observability-Tools ein, einschließlich Analyse und Protokollierung. Siehe [Überwachung von Amazon GameLift Servers](#).

Entwicklungs-Roadmap für Hybrid-Hosting mit Amazon GameLift Servers

Diese Roadmap führt dich durch die Entwicklung einer Hosting-Lösung für dein Multiplayer-Spiel. Amazon GameLift Servers bietet mehrere Hosting-Optionen für Spiele. Weitere Informationen zu diesen Optionen finden Sie unter [Amazon GameLift Servers Hosting-Optionen für Spiele](#).

Eine Hybridlösung verwendet eine Kombination aus Hosting-Ressourcen, einschließlich cloudbasierter Ressourcen, die von verwaltet werden, Amazon GameLift Servers und Ihrer eigenen, selbst verwalteten Hosting-Ressourcen. Eine ausführlichere Erläuterung des Hybrid-Hostings finden Sie in diesem Artikel: [Hybrid-GameServer-Hosting mit Amazon GameLift Servers Anywhere](#). Mit Amazon GameLift Servers können Sie eine Hybridlösung einrichten, die gemeinsame Komponenten und Prozesse verwendet, sodass Sie eine globale Flotte zentral verwalten und Lasten problemlos zwischen allen Arten von Ressourcen verschieben können.

Eine Hybridarchitektur besteht aus den folgenden Komponenten:

- Ein einziger Spieleserver-Build, der in das Server-SDK für integriert ist Amazon GameLift Servers und auf allen Flotten eingesetzt werden kann.
- Ein einziger Spielclient und Backend-Service, integriert in das AWS SDK, um mit dem Amazon GameLift Servers Dienst zu interagieren und Spielsitzungen anzufordern.
- Eine gemeinsame Amazon GameLift Servers Warteschlange, um neue Spielsitzungen auf verfügbaren Spieleservern zu platzieren und die Auslastung auf alle Flotten verteilen zu können.
- (Optional) Ein Process Manager-Agent, der zusammen mit einer Anywhere-Flotte bereitgestellt wird, vereinfacht die Aufgaben des Serverprozessmanagements für alle Computer in allen Flotten.
- (Optional) Ein FlexMatch Matchmaker, um Spiele für mehrere Spieler zu erstellen und Spielsitzungen für sie einzurichten.
- Eine oder mehrere Amazon GameLift Servers verwaltete Flotten, die Amazon Elastic Compute Cloud (Amazon EC2) -Instances verwenden, die für das Hosten von Spielen im Mehrspielermodus optimiert sind.

- Eine oder mehrere Amazon GameLift Servers Anywhere-Flotten, die Ihre vorhandenen lokalen oder anderen Hosting-Ressourcen nutzen, einschließlich Ihrer Tools für das Konfigurationsmanagement und die Bereitstellung. (Sie können optional die verwenden.) AWS Systems Manager

Diese Roadmap bietet einen optimierten Weg, um Ihr Multiplayer-Spiel in einer Hybrid-Hosting-Lösung mit erfolgreich zum Laufen zu bringen. Amazon GameLift Servers Sobald Sie die erforderlichen Komponenten installiert haben, können Sie mit der Spieleentwicklung fortfahren und Ihre Hosting-Lösung anpassen. Je näher die Markteinführung rückt, finden Sie hier Hilfe bei der Vorbereitung Ihrer Hosting-Lösung für den Einsatz auf Produktionsebene. [Bereite dich mit Amazon GameLift Servers Hosting auf den Start vor](#)

 Holen Sie sich eine Starthilfe mit dem Plugin Amazon GameLift Servers

Wenn du Projekte mit Unreal Engine oder Unity entwickelst, beginne mit der Einrichtung deines Spiels für das Hosting mit dem Amazon GameLift Servers Plugin. Mit dem Plugin kannst du es Amazon GameLift Servers SDKs zu deinem Spielprojekt hinzufügen und mithilfe der geführten Workflows eine einfache, funktionierende Version einer Hybrid-Hosting-Lösung mit sowohl einer Anywhere-Flotte als auch einer Amazon GameLift Servers verwalteten Flotte erstellen. Anschließend kannst du auf diesen Grundlagen aufbauen und sie nach Bedarf anpassen.

Schritt 1: Bereite deinen Spieleserver für die Arbeit vor Amazon GameLift Servers

Füge deinem Spieleserver Funktionen hinzu, sodass er mit dem Amazon GameLift Servers Dienst kommunizieren kann, wenn er zum Hosten bereitgestellt wird. Dieselbe Funktionalität ist erforderlich, wenn der Spieleserver auf einer Amazon GameLift Servers verwalteten Flotte oder einer Anywhere-Flotte läuft.

- Holen Sie sich das Server-SDK für Amazon GameLift Servers (Version 5.x) für Ihr Spielprojekt. Das Server-SDK ist in C++, C# und Go verfügbar. [Laden Sie das Server-SDK für Amazon GameLift Servers](#) herunter.
- Ändern Sie den Code Ihres Spieleservers, um Server-SDK-Funktionen hinzuzufügen. Anleitungen finden Sie unter [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#). Gehen Sie mindestens wie folgt vor:

- Fügen Sie Code hinzu, um das Amazon GameLift Servers SDK zu initialisieren und eine WebSocket Verbindung mit dem Amazon GameLift Servers Dienst herzustellen. Verwenden Sie die Server-SDK-Aktion `InitSdk()`. Fügen Sie Code zur Angabe von Serverparametern bei der Ausführung auf einem Anywhere-Flottencomputer hinzu.
- Fügen Sie Code hinzu, um dem Amazon GameLift Servers Dienst zu melden, wenn der Serverprozess bereit ist, Spielsitzungen abzuhalten. Verwenden Sie die Server-SDK-Aktion `ProcessReady()`.
- Implementieren Sie die erforderlichen `OnProcessTerminate()` Rückruffunktionen und `OnStartGameSession()`. Mit diesen Funktionen können Spielserverprozesse eine Verbindung mit dem Amazon GameLift Servers Dienst aufrechterhalten, auf Aufforderung von Amazon GameLift Servers eine Spielsitzung einleiten und auf eine Aufforderung zum Beenden des Spielserverprozesses reagieren.
- Fügen Sie Code hinzu, um dem Amazon GameLift Servers Dienst zu melden, wenn der Serverprozess eine Spielsitzung beendet. Verwenden Sie die Server-SDK-Aktion `ProcessEnding()`.
- Verpacke deinen Gameserver-Build. Erstelle ein Installationsskript mit deinen Build-Dateien, Abhängigkeiten und unterstützender Software. Siehe [Packen Ihrer Spielserver-Build-Dateien](#). Wir empfehlen, einen Amazon Simple Storage Service (Amazon S3) -Bucket zu verwenden, um Versionen Ihres Game-Builds zu speichern.
- Testen Sie die Integration Ihres Spielervers. Für diese Aufgabe empfehlen wir, eine Amazon GameLift Servers Anywhere-Flotte mit einer lokalen Workstation einzurichten, wie unter [beschrieben Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#). Installieren Sie für diesen Schritt Ihren Gameserver-Build manuell auf dem Testgerät und starten Sie einen Serverprozess. Verwenden Sie die AWS CLI, um eine neue Spielsitzung anzufordern, und stellen Sie sicher, dass der Amazon GameLift Servers Dienst Ihren Serverprozess erfolgreich zum Starten einer Spielsitzung auffordert.

Schritt 2: Bereite deinen Spielclient für die Teilnahme an gehosteten Spielsitzungen vor

Erstelle eine Möglichkeit für deinen Spielclient, die Teilnahme an einer Spielsitzung anzufordern, Verbindungsinformationen abzurufen und dann direkt eine Verbindung zu einer gehosteten Spielsitzung herzustellen. Der gängigste Ansatz besteht darin, eine Backend-Servicefunktion einzurichten, die als Vermittler zwischen deinem Spielclient und dem Dienst dient. Amazon GameLift Servers Dieser Ansatz schützt deine Hosting-Ressourcen und gibt dir mehr Kontrolle darüber, wie Spieler an Spielsitzungen teilnehmen.

- Entwickeln Sie Backend-Servicefunktionen für das Hosting. Der Backend-Dienst kommuniziert mit dem Amazon GameLift Servers Dienst und übermittelt Verbindungsinformationen an einen Spieleclient. Diese Funktion umfasst das Starten von Spielsitzungen, das Platzieren von Spielern in Spiele und das Abrufen von Informationen zu Spielsitzungen. Anleitungen finden Sie unter [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#). Gehen Sie mindestens wie folgt vor:
 - Holen Sie sich das AWS SDK für Amazon GameLift Servers und fügen Sie es zu Ihrem Backend-Serviceprojekt hinzu. Siehe [Amazon GameLift Servers SDK-Ressourcen für Kundendienste](#).
 - Fügen Sie Code hinzu, um einen Amazon GameLift Servers Client zu initialisieren und wichtige Einstellungen zu speichern. Siehe [Richten Sie die Amazon GameLift Servers API ein](#).
 - Fügen Sie Funktionen hinzu, um die AWS SDK-Aktion aufzurufen `CreateGameSession()` und einem Spielclient Verbindungsinformationen für die Spielsitzung bereitzustellen. Weitere Informationen [findest du unter Eine Spielsitzung auf einer bestimmten Flotte erstellen](#).

Ein Anruf `CreateGameSession()` ist ein praktischer Ausgangspunkt, um neue Spielsitzungen anzufordern. Sobald du ein System zur Platzierung von Spielsitzungen eingerichtet hast (siehe Schritt 3), ersetzt du diesen Code durch einen Anruf an `StartGameSessionPlacement()` (oder, `StartMatchmaking()` falls du es verwendest `FlexMatch`).

Anleitungen zur Gestaltung Ihres Back-End-Dienstes finden Sie unter [Erstellen Sie einen Backend-Service für Amazon GameLift Servers](#).

- Füge deinem Spielclient Funktionen hinzu, mit denen Spieler an einer gehosteten Spielsitzung teilnehmen können. Der Spielclient sendet Anfragen an deinen Backend-Service, nicht direkt an Amazon GameLift Servers. Nachdem der Backend-Dienst Verbindungsinformationen zur Spielsitzung bereitgestellt hat, stellt der Spielclient eine direkte Verbindung mit der Spielsitzung her, um das Spiel zu spielen.
- Testen Sie die Integration Ihres Spielclients. Sie können dieselbe Amazon GameLift Servers Anywhere-Flotte mit einer lokalen Workstation zum Testen verwenden.

Wenn Sie während der Entwicklungsphase testen möchten, wie sich Ihr Spielbuild in einer Amazon GameLift Servers verwalteten Flotte verhält, empfehlen wir Ihnen, auch eine [cloudbasierte Testumgebung](#) einzurichten. Diese Amazon GameLift Servers Toolkit-Lösung ahmt das Verhalten einer verwalteten Flotte nach, ermöglicht es Ihnen jedoch, Spielserver-Builds mit minimaler Bearbeitungszeit zu aktualisieren.

Schritt 3: Richten Sie die Platzierung der Spielsitzungen ein

Passen Sie an Amazon GameLift Servers, wie Sie Anfragen für neue Spielsitzungen bearbeiten möchten, und suchen Sie nach verfügbaren Spieleservern, auf denen sie gehostet werden können. Amazon GameLift Servers verfolgt automatisch die Verfügbarkeit aller Spieleserver auf allen Flotten. Wenn ein Spielclient eine Anfrage zur Teilnahme an einer Spielsitzung sendet, Amazon GameLift Servers sucht er anhand definierter Prioritäten wie minimale Latenz, Kosten und Verfügbarkeit nach der „bestmöglichen“ Platzierung.

- Erstelle eine Warteschlange für eine Spielsitzung, um eine neue Spielsitzung auf verfügbaren Spieleservern zu platzieren. Warteschlangen sind der wichtigste Mechanismus für die Platzierung von Spielsitzungen. Anleitungen finden Sie unter [Eine Warteschlange für Spielsitzungen erstellen](#).
- Fügen Sie Ihrer Warteschlange mindestens Ihre Anywhere-Flotten als Ziele hinzu. Alle anderen Einstellungen sind optionale Anpassungen.
- Konvertieren Sie den **CreateGameSession()** Anruf in Ihrem Backend-Servicecode in **StartGameSessionPlacement()**. Weitere Informationen findest du unter [Eine Spielsitzung in einer Warteschlange mit mehreren Standorten erstellen](#).
- Erstelle einen Mechanismus, um einen Spielclient zu benachrichtigen, wenn eine Spielsitzung bereit ist, daran teilzunehmen. Während der Entwicklung kannst du den Status einer Spielsitzung abfragen, indem du einen Call to verwendest `DescribeGameSessionPlacement`. Bevor du jedoch eine Warteschlange zur Verarbeitung großer Datenmengen verwenden kannst, musst du die Ereignisbenachrichtigungen aktivieren. Siehe [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#).
- (Optional) Fügen Sie FlexMatch Matchmaking-Komponenten hinzu. Anleitungen finden Sie im [Amazon GameLift Servers FlexMatch Entwicklerhandbuch](#).

Schritt 4: Richten Sie mit dem Amazon GameLift Servers Agenten eine Anywhere-Flotte ein

Bis zu diesem Zeitpunkt hast du mit lokalen Geräten (registriert als Anywhere Fleet Computes) gearbeitet, um deine Spielkomponenten zu testen und zu verbessern. Der nächste Schritt besteht darin, die Art von Flotten einzurichten, die Sie für ein Produktionssystem benötigen. Beginnen Sie mit einer Anywhere-Flotte und fügen Sie den Amazon GameLift Servers Agenten hinzu, um einige wichtige Aufgaben zur Serververwaltung am Computer zu verwalten. Weitere Details finden Sie unter [Arbeite mit dem Amazon GameLift Servers Agenten](#).

- Hol dir den Amazon GameLift Servers Agenten und füge ihn deinem Gameserver-Installationspaket hinzu. Holen Sie sich den Agent-Quellcode, der im [Github-Repository für Amazon GameLift](#)

[Servers Agenten verfügbar ist, und erstellen Sie ihn.](#) Platzieren Sie die resultierende ausführbare JAR-Datei in demselben Verzeichnis wie die ausführbare Datei für den Spiel-Build.

- Ändern Sie Ihr Startskript für den Agenten nach Bedarf. Stellen Sie sicher, dass die ausführbare Datei des Agenten gestartet wird, sobald ein Computer ausgeführt wird. Hilfe zur Installation und Ausführung des Agenten auf Ihren Hosting-Computern finden Sie in der Readme-Datei im Agent-Repository. Ihr Startbefehl sollte Optionen enthalten, mit denen Sie mindestens die Anywhere-Flotten-ID sowie einen benutzerdefinierten Standort und AWS-Region einen Computernamen angeben können.

Der Agent erledigt automatisch die folgenden Aufgaben für Sie. Wenn Sie diese Aufgaben also mit Skripten erledigt haben, können Sie sie entfernen:

- Ruft `RegisterCompute()` auf, um die Rechenleistung einer Anywhere-Flotte hinzuzufügen.
- Aufrufe `GetComputeAuthToken()` zur Authentifizierung von Spieleservern, wenn sie sich mit dem Amazon GameLift Servers Dienst verbinden. Der Agent verwaltet das Abrufen und Aktualisieren des Authentifizierungstokens, das von allen Spieleserverprozessen verwendet werden kann, die auf dem Computer ausgeführt werden.
- Startet neue Serverprozesse auf dem Computer auf der Grundlage einer Reihe von Laufzeitanweisungen.
- Erstellen Sie eine Laufzeitkonfiguration für Computer in Ihrer Anywhere-Flotte. Geben Sie mindestens den Startpfad für die ausführbare Datei Ihres Spieleservers an. Sie können die Amazon GameLift Servers Konsole oder die AWS CLI verwenden, um Laufzeitanweisungen für die Flotte zu erstellen oder zu ändern. Der Agent führt diese Anweisungen aus und fordert regelmäßig Updates vom Amazon GameLift Servers Service an.
- Richten Sie Ihre Warteschlange für Spielsitzungen nach Bedarf ein oder ändern Sie sie. Erstellen Sie eine neue Warteschlange (oder aktualisieren Sie eine bestehende) und legen Sie ein Ziel für die Anywhere-Flotte fest, die zusammen mit dem Amazon GameLift Servers Agenten eingesetzt wird.
- Testen Sie die Agent-Integration mit Ihren Anywhere-Flotten. Stellen Sie sicher, dass der Agent die Serverprozesse auf der Grundlage der Laufzeitkonfiguration ordnungsgemäß startet.

Schritt 5: Erstellen Sie eine verwaltete Cloud-basierte Flotte

Erstellen Sie eine Amazon GameLift Servers verwaltete EC2-Flotte als Ergänzung zu Ihrer Anywhere-Flotte. Wenn Sie in Schritt 2 eine cloudbasierte Testumgebung eingerichtet haben, um die Entwicklung zu beschleunigen, sollten Sie planen, eine verwaltete Flotte einzurichten, nachdem Sie den Großteil Ihrer Spieleentwicklung und Tests abgeschlossen haben. Du benötigst eine

vollständig verwaltete Flotte, um zusätzliche Einstellungen wie die automatische Kapazitätsskalierung konfigurieren und testen zu können.

- Package Sie Ihren Gameserver-Build und laden Sie ihn hochAmazon GameLift Servers. Erstelle ein Installationsskript mit deinen Build-Dateien, Abhängigkeiten und unterstützender Software. Sie können dieselbe Build-Software sowohl für Ihre Anywhere- als auch für Ihre verwalteten Flotten verwenden. Siehe [Erstelle einen Gameserver-Build für Amazon GameLift Servers](#). Sie können Ihren Build entweder Amazon GameLift Servers über die Konsole oder die AWS CLI hochladen.

Bevor Sie Ihren Build hochladen, entscheiden Sie, in welchem AWS-Region Bereich Sie die verwaltete Flotte erstellen möchten. Sie müssen den Build in dieselbe Region hochladen. Weitere Informationen zur Auswahl eines Flottenstandorts finden Sie unter[Geografische Standorte](#).

- Erstellen Sie eine verwaltete EC2-Flotte. Sie können die Amazon GameLift Servers Konsole oder die AWS CLI verwenden, um eine verwaltete Flotte zu erstellen. Wenn du eine Flotte erstellst, wird Amazon GameLift Servers sofort mit der Bereitstellung deines Gameserver-Builds für das Hosting begonnen. Du kannst viele Aspekte einer verwalteten Flotte konfigurieren. Anleitungen finden Sie unter [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#). Gehen Sie mindestens wie folgt vor:
 - Gib der Flotte einen Namen und gib an, welcher hochgeladene Spiel-Build eingesetzt werden soll.
 - Wählen Sie On-Demand-Instances für Ihre Flotte und wählen Sie einen Instance-Typ aus, der an Ihrem Flottenstandort verfügbar ist. Spot-Flotten sind eine wertvolle Option, erfordern jedoch zusätzliches Design und Konfiguration.
 - Erstellen Sie eine Runtime-Konfiguration mit ähnlichen Einstellungen wie bei der Anywhere-Flotte. Geben Sie mindestens den Startpfad für die ausführbare Datei Ihres Gameservers an.
 - Geben Sie die Porteneinstellungen an, damit eingehender Datenverkehr auf Ihre Spieleserver zugreifen kann.
- Füge die verwaltete Flotte zu deiner Warteschlange für gemeinsame Spielsitzungen hinzu. Aktualisiere die Warteschlange aus Schritt 4, sodass sie Ziele sowohl für die verwaltete Flotte als auch für die mit dem Amazon GameLift Servers Agenten eingesetzte Anywhere-Flotte enthält.
- Testen Sie das Spiele-Hosting mit Ihren verwalteten Flotten. Zu diesem Zeitpunkt solltest du in der Lage sein, den gesamten Hosting-Zyklus zu testen. Ein Spieleclient fordert eine Spielsitzung an, ruft Verbindungsinformationen ab und stellt erfolgreich eine Verbindung zu einer Spielsitzung her.

Schritt 6: Passen Sie Ihre Flotten an

Während du dich auf den Start des Spiels vorbereitest, musst du deine Hosting-Lösungen verfeinern. Zu den zu berücksichtigenden Entscheidungen gehören unter anderem:

- Automatisieren Sie bei Anywhere-Flotten den Prozess des Startens und Herunterfahrens von Computern nach Bedarf, einschließlich der Installation und Ausführung von Gameserver-Software. Das Wiederverwenden von Computern ist nützlich, um sicherzustellen, dass sie regelmäßig aktualisiert werden. Durch das Herunterfahren von Computern können Kosten eingespart werden, wenn sie nicht benötigt werden.
- Wenn dein Spieleserver andere AWS Ressourcen kommunizieren muss, richte IAM-Rollen ein, um den Zugriff zu verwalten. Siehe [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).
- Ermitteln Sie, wo Sie die Spieleserver geografisch platzieren möchten. Fügen Sie Ihren verwalteten Flotten entfernte Standorte hinzu. Siehe [Anpassungen der Hosting-Ressourcen](#).
- Erwägen Sie bei verwalteten Flotten den Einsatz von Spot-Flotten, um Kosten zu sparen. Siehe [Senken Sie die Kosten für das Hosting von Spielen mit Spot-Flotten](#).
- Optimieren Sie die Flottenleistung, indem Sie Konfigurationen für Rechenressourcen auswählen und anschließend die Runtime-Anweisungen so konfigurieren, dass die optimale Anzahl von Serverprozessen pro Recheneinheit ausgeführt wird. Tun Sie dies sowohl für Anywhere-Flotten als auch für verwaltete Flotten. Siehe [Optimiere die Laufzeitkonfiguration des Spieleservers auf verwalteten Amazon GameLift Servers](#).
- Experimentieren Sie mit den Optionen zur Platzierung von Spielsitzungen für verwaltete Flotten, einschließlich der Anpassung der Priorisierungseinstellungen. Siehe [Passen Sie eine Warteschlange für Spielsitzungen an](#).
- Richten Sie für verwaltete Flotten die automatische Kapazitätsskalierung ein, um der erwarteten Spielernachfrage gerecht zu werden. Siehe [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#).
- Richten Sie für Anywhere-Flotten Mechanismen ein, die eine manuelle oder automatische Kapazitätsskalierung ermöglichen, um der erwarteten Nachfrage der Spieler gerecht zu werden.
- Entwerfen und implementieren Sie bei Bedarf einen Failover auf andere Ressourcen. Richten Sie Standby-Flotten in anderen Flotten in anderen AWS-Regionen und ändern Sie Warteschlangen und auto Skalierung, um bei Bedarf Failover abzuwickeln.

- Richten Sie Hosting-Observability-Tools ein, einschließlich Analyse und Protokollierung. Siehe [Überwachung von Amazon GameLift Servers](#). Erstellen Sie Metrikgruppen, um Analysen für all Ihre Hosting-Ressourcen zu aggregieren.
- Automatisieren Sie Ihre Bereitstellung mithilfe von [Infrastructure as Code \(IaC\)](#). Siehe [Amazon GameLift Servers Hosting-Ressourcen verwalten mit CloudFormation](#).

Amazon GameLift Servers unterstützt die Verwendung von AWS CloudFormation Vorlagen für beliebige bereitstellungsspezifische Konfigurationen. Sie können die auch verwenden, um Ihre AWS Cloud Development Kit (AWS CDK) Amazon GameLift Servers Ressourcen zu definieren. Weitere Informationen zu finden Sie im [AWS Cloud Development Kit \(AWS CDK\) Entwicklerhandbuch](#). AWS CDK

Um die Bereitstellung Ihrer CloudFormation Stacks zu verwalten, empfehlen wir die Verwendung von Tools und Diensten für Continuous Integration und Continuous Delivery (CI/CD), wie z. AWS CodePipeline Mit diesen Tools können Sie die Installation automatisch oder mit Genehmigung durchführen, wann immer Sie eine Binärdatei für einen Spielservers erstellen. Mit einem CI/CD Tool oder Dienst kann die Bereitstellung von Ressourcen für eine neue Spielserversversion wie folgt aussehen:

- Erstelle und teste deine Gameserver-Binärdatei.
- Lade die Binärdatei auf hochAmazon GameLift Servers.
- Stellen Sie mit dem neuen Build neue Flotten bereit.
- Füge die neuen Flotten zu deiner Warteschlange für Spielsitzungen hinzu und entferne die Flotten mit der vorherigen Build-Version.
- Wenn die Flotten mit dem vorherigen Build keine aktiven Spielsitzungen mehr veranstalten, lösche die CloudFormation Stapel dieser Flotten.

Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers

Bevor du dein Spiel zum Hosten bereitstellen kannst Amazon GameLift Servers, musst du sowohl die Komponenten deines Spielservers als auch deines Spielclients vorbereiten. Dazu gehört die Integration deines Spielservers in das Amazon GameLift Servers Server-SDK und der Aufbau eines Backend-Dienstes, der die Spielerauthentifizierung und die Verwaltung von Spielsitzungen übernimmt.

Die Server-SDK-Integration ermöglicht es deinem Spielservers, mit dem Amazon GameLift Servers Dienst zu kommunizieren, seinen Status zu melden und Spielsitzungen zu verwalten. Der Backend-

Dienst fungiert als Vermittler zwischen deinen Spielclients und dem Amazon GameLift Servers Dienst, bearbeitet Spielanfragen und koordiniert die Platzierung der Spielsitzungen.

Themen

- [Bereite dein Unreal- oder Unity-Spiel mit dem Amazon GameLift Servers Plugin vor](#)
- [Integrieren Sie einen Spieleserver mit Amazon GameLift Servers](#)
- [Einen Gameserver-Build für die Bereitstellung verpacken](#)
- [client/server Spielinteraktionen mit Amazon GameLift Servers](#)

Bereite dein Unreal- oder Unity-Spiel mit dem Amazon GameLift Servers Plugin vor

Das Amazon GameLift Servers Plugin ist ein Add-On mit vollem Funktionsumfang für deine Unreal- oder Unity-Game-Engine. Es führt dich durch die grundlegenden Schritte zur Bereitstellung deines Spiels für das Hosting. Amazon GameLift Servers Mit den Tools und Workflows des Plugins kannst du in deiner Game-Engine-Entwicklungsumgebung arbeiten, um deinen Spieleserver für das Hosting vorzubereiten, das Hosting auf einem lokalen Computer zum Testen einzurichten, einen einfachen Backend-Service zu erstellen und deinen Spieleserver für verwaltetes Cloud-basiertes Hosting bereitzustellen.

Nutze das Plugin, um Erfahrungen mit der Arbeit mit Amazon GameLift Servers einer Game-Hosting-Lösung zu sammeln und sie schnell zum Laufen zu bringen. Du kannst mit Beispiel-Spiel-Assets oder deinem eigenen Spielprojekt arbeiten. Das Plugin automatisiert eine Reihe von Schritten, sodass Sie schnell eine einfache, funktionierende Lösung erstellen können. Wenn du die geführten Workflows des Plugins abgeschlossen hast, kannst du über Amazon GameLift Servers einen Spielclient eine Verbindung zu live gehosteten Spielsitzungen herstellen. Nachdem du das Plugin verwendet hast, um eine einfache Hosting-Lösung zu erstellen, kannst du deine Lösung an die Bedürfnisse deines Spiels anpassen.

Das Plugin ist für die folgenden Game-Engines verfügbar:

- Unreal Engine
- Unity

Das Plugin enthält diese Komponenten für jede Spiel-Engine:

- Plugin-Module für den Game-Engine-Editor. Wenn das Plugin installiert ist, erhalten Sie über eine neue Hauptmenüschaltfläche Zugriff auf die Amazon GameLift Servers Funktionen.
- Bibliotheken für die Amazon GameLift Servers Service-API mit clientseitiger Funktionalität.
- Bibliotheken für das Amazon GameLift Servers Server-SDK (Version 5).
- Beispielressourcen zum Testen einer Serverintegration.
- Bearbeitbare Konfigurationen in Form von CloudFormation Vorlagen, die Ihre Spieleserverlösung definieren.

Themen

- [Plugin-Arbeitsablauf](#)
- [Amazon GameLift ServersPlugin für Unreal Engine](#)
- [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#)
- [Amazon GameLift ServersPlugin für Unity für Server SDK 4](#)

Plugin-Arbeitsablauf

Die folgenden Schritte beschreiben einen typischen Weg zur Vorbereitung und Bereitstellung Ihres Spieleprojekts auf Amazon GameLift Servers. Sie führen diese Schritte durch, indem Sie im Game-Engine-Editor und in Ihrem Spielcode arbeiten.

1. Erstellen Sie ein Benutzerprofil, das auf Ihren AWS Kontobenzutzer verweist und Zugangsdaten mit Nutzungsberechtigungen bereitstellt Amazon GameLift Servers.
2. Richten Sie verwandte AWS Ressourcen ein, die das Plugin in der Hosting-Lösung verwendet (als „Bootstrapping“ bezeichnet).
3. Füge deinem Projekt Servercode hinzu, um die Kommunikation zwischen einem laufenden Spieleserver und dem Amazon GameLift Servers Dienst herzustellen.
4. Füge deinem Projekt einen Client-Code hinzu, mit dem Spieleclients Anfragen Amazon GameLift Servers zum Starten neuer Spielsitzungen senden und dann eine Verbindung zu ihnen herstellen können.
5. Verwenden Sie den Anywhere-Workflow, um Ihre lokale Workstation als Anywhere-Computer einzurichten und Ihren Spieleserver zu hosten. Starte deinen Gameserver und Client lokal über das Plugin, stelle eine Verbindung zu einer Spielsitzung her und teste die Integration.
6. Verwende den verwalteten EC2 Workflow, um deinen Spieleserver auf eine einfache, aber vollständige Cloud-Hosting-Lösung hochzuladen Amazon GameLift Servers und bereitzustellen.

Starte deinen Spielclient lokal über das Plugin, fordere eine Spielsitzung an, stelle eine Verbindung her und spiele dein Spiel.

Wenn du mit dem Plugin arbeitest, erstellst und AWS verwendest du Ressourcen. Für diese Aktionen können Gebühren für das verwendete AWS Konto anfallen. Wenn Sie noch nicht damit vertraut sind AWS, fallen diese Aktionen möglicherweise unter das [AWS kostenlose Kontingent](#).

Amazon GameLift ServersPlugin für Unreal Engine

Dieses Plugin fügt dem UE-Editor das Amazon GameLift Servers C++-Server-SDK und die Tools hinzu. Verwenden Sie die geführten UI-Workflows, um die Server-SDK-Funktionalität in Ihr Spieleprojekt zu integrieren und eine Amazon GameLift Servers Hosting-Lösung für Ihren Spieleserver bereitzustellen.

Mit dem Plugin kannst du eine grundlegende, funktionierende Hosting-Lösung erstellen und diese dann nach Bedarf optimieren und anpassen. Richten Sie eine Amazon GameLift Servers Anywhere-Flotte mit Ihrer lokalen Workstation als Host ein. Stellen Sie für Cloud-Hosting mit verwalteten EC2- oder verwalteten Container-Flotten Ihren Spieleserver mit einer Komplettlösung zur Verwaltung von Spielsitzungsanfragen und Client-Verbindungen bereit.

Themen

- [Installiere das Plugin für dein Unreal-Spieleprojekt](#)
- [Nächste Schritte: Passe deine Game-Hosting-Lösung an](#)
- [Plugin für Unreal: Richten Sie ein AWS Benutzerprofil ein](#)
- [Plugin für Unreal: Integriere deinen Spielcode](#)
- [Plugin für Unreal: Hoste dein Spiel lokal mit Anywhere Amazon GameLift Servers](#)
- [Plugin für Unreal: Stellen Sie Ihr Spiel auf einer verwalteten EC2-Flotte bereit](#)
- [Plugin für Unreal: Setze dein Spiel auf einer verwalteten Containerflotte ein](#)

Installiere das Plugin für dein Unreal-Spieleprojekt

[Hol dir das Amazon GameLift Servers Plugin für Unreal Engine von GitHub](#)

In der GitHub Readme-Datei des Repositorys findest du Informationen darüber, wie du das Plugin in deinem Unreal Editor für ein Spieleprojekt installierst.

Das Plugin beinhaltet die folgenden Komponenten:

- Plugin-Module für den UE-Editor. Wenn das Plugin installiert ist, erhalten Sie über eine neue Hauptmenüschaltfläche Zugriff auf die Amazon GameLift Servers Funktionen.
- C++-Bibliotheken für die Amazon GameLift Servers Service-API. Verwenden Sie die API-Funktionalität in einem clientseitigen Backend-Dienst, um Spieleclients beim Anfordern von Spielsitzungen und send/retrieve Informationen zu Spielsitzungen zu unterstützen.
- Unreal-Bibliotheken für das Amazon GameLift Servers Server-SDK (Version 5). Verwende das Server-SDK in deinem Gameserver-Code, um die Kommunikation zwischen gehosteten Spieleserverprozessen und dem Amazon GameLift Servers Dienst zu verwalten.
- Inhalte zum Testen, darunter eine Startkarte für das Spiel und zwei Testkarten mit grundlegenden Entwürfen und Benutzeroberflächenelementen zum Testen einer Serverintegration.
- Bearbeitbare Konfigurationen in Form von CloudFormation Vorlagen, die das Plugin bei der Bereitstellung deines Spieleservers für das Hosting verwendet.

Dieses Plugin verwendet AWS CloudFormation Vorlagen, um Hosting-Lösungen für gängige Spieleszenarien bereitzustellen. Verwenden Sie diese Lösungen wie bereitgestellt oder passen Sie sie nach Bedarf für Ihre Spiele an.

Nächste Schritte: Passe deine Game-Hosting-Lösung an

Die Verwendung der geführten Workflows des Plugins ist eine gute Möglichkeit, mit einer Amazon GameLift Servers Hosting-Lösung schnell loszulegen. Mit dem Plugin können Sie Basisversionen der einzelnen Komponenten Ihrer Lösung einrichten.

Wenn du bereit bist, kannst du auf dieser Basislösung aufbauen, indem du jede Komponente individuell anpasst und deine Lösung verfeinerst, während du dich auf den Spielstart vorbereitest. Ziehe diese Optionen in Betracht:

- Ändern Sie Ihre Flotten und Flottenkonfigurationen. Siehe [Anpassungen der Hosting-Ressourcen](#).
- Passen Sie die Konfiguration Ihrer Warteschlange für Spielsitzungen an. Siehe [Passen Sie eine Warteschlange für Spielsitzungen an](#):
- Füge deinem Spieleserver und Spielclient Funktionen hinzu. Siehe [Integrieren Sie einen Spieleserver mit Amazon GameLift Servers](#) und [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#).
- Passen Sie Ihren Backend-Service an. Siehe [Erstellen Sie einen Backend-Service für Amazon GameLift Servers](#).

- Richten Sie die automatische Kapazitätsskalierung ein, um der erwarteten Spielernachfrage gerecht zu werden. Siehe [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#).
- Richten Sie Hosting-Observability-Tools ein, einschließlich Analyse und Protokollierung. Siehe [Überwachung von Amazon GameLift Servers](#).
- Automatisieren Sie Ihre Bereitstellung mithilfe von [Infrastructure as Code \(IaC\)](#). Die geführten Workflows des Plugins für verwaltete Lösungen verwenden AWS CloudFormation Vorlagen. Sie können diese nach Bedarf anpassen. Siehe [Amazon GameLift Servers Hosting-Ressourcen verwalten mit CloudFormation](#).

Themen

- [Plugin für Unreal: Richten Sie ein AWS Benutzerprofil ein](#)
- [Plugin für Unreal: Integriere deinen Spielcode](#)
- [Plugin für Unreal: Hoste dein Spiel lokal mit Anywhere Amazon GameLift Servers](#)
- [Plugin für Unreal: Stellen Sie Ihr Spiel auf einer verwalteten EC2-Flotte bereit](#)
- [Plugin für Unreal: Setze dein Spiel auf einer verwalteten Containerflotte ein](#)

Plugin für Unreal: Richten Sie ein AWS Benutzerprofil ein

Richten Sie nach der Installation des Plugins ein Benutzerprofil mit einem gültigen AWS Konto ein. Sie können mehrere Profile im Plugin verwalten, aber Sie können jeweils nur ein Profil auswählen. Wann immer Sie im Plugin arbeiten, wählen Sie ein Profil aus, das Sie verwenden möchten. Auf jeder Workflow-Seite wird das aktuell ausgewählte Profil angezeigt.

Wenn Sie mehrere Profile verwalten, können Sie zwischen verschiedenen Hosting-Bereitstellungen wechseln. Sie können beispielsweise Profile einrichten, die dasselbe AWS Konto verwenden, aber in verschiedenen AWS Regionen bereitgestellt werden. Oder Sie können Profile mit unterschiedlichen AWS Konten oder Benutzern und Berechtigungssätzen einrichten.

Note

Wenn Sie die AWS CLI auf Ihrer Workstation installiert haben und bereits ein Profil konfiguriert haben, erkennt das Amazon GameLift Servers Plugin es und listet es als vorhandenes Profil auf. Das Plugin wählt automatisch jedes benannte Profil aus[default]. Sie können ein vorhandenes Profil verwenden oder ein neues erstellen.

Für alle Profile muss ein Bootstrapping durchgeführt werden, um einige erforderliche AWS Ressourcen unter Ihrem Kontonutzer einzurichten.

Um Ihre Profile zu verwalten AWS

1. Wählen Sie in der Hauptwerkzeugleiste des Unreal Editors das Amazon GameLift Servers Menü und dann AWS Access Credentials aus. Diese Aktion öffnet das Amazon GameLift Servers Plugin auf der Seite Richten Sie Ihr Benutzerprofil ein.
2. Verwenden Sie die Schaltflächen, um ein neues AWS Konto zu erstellen oder ein Benutzerprofil für ein AWS Konto einzurichten, das Sie bereits haben.
3. Wenn Sie noch kein Benutzerprofil haben, werden Sie aufgefordert, Profildetails einzugeben und ein neues Profil zu erstellen. Geben Sie die folgenden Informationen ein:
 - Ein AWS Konto. Wenn Sie ein neues AWS Konto erstellen, verwenden Sie den Link zum AWS-Managementkonsole und folgen Sie den Anweisungen. Weitere Informationen finden [Sie AWS unter Konto erstellen](#).
 - Ein AWS Benutzer mit Nutzungsberechtigungen Amazon GameLift Servers und anderen erforderlichen AWS Diensten. Anweisungen [Richten Sie ein AWS Benutzerkonto ein](#) zum Einrichten eines AWS Identity and Access Management (IAM-) Benutzers mit Amazon GameLift Servers Berechtigungen und programmatischem Zugriff mit langfristigen Anmeldeinformationen finden Sie unter.
 - Anmeldeinformationen für Ihren AWS Benutzer. Diese Anmeldeinformationen bestehen aus einer AWS Zugriffsschlüssel-ID und einem AWS geheimen Schlüssel. Weitere Informationen finden Sie unter [Holen Sie sich Ihre Zugangsschlüssel](#).
 - AWS Region. Dies ist ein geografischer Standort, an dem Sie Ihre AWS Ressourcen für das Hosting erstellen möchten. Wir empfehlen, während der Entwicklung eine Region in der Nähe deines physischen Standorts zu verwenden. Wählen Sie eine Region aus der Liste der [unterstützten AWS Regionen](#) aus.
4. Wenn das Plugin vorhandene Profile erkennt, zeigt es eine Liste der verfügbaren Profile an. Wählen Sie ein vorhandenes Profil aus der Liste aus oder wählen Sie Weiteres Profil hinzufügen, um ein neues zu erstellen.

Bootstrap für ein Benutzerprofil

Alle Profile müssen bootstrapped sein, um sie mit dem Plugin verwenden zu können. Amazon GameLift Servers Bootstrapping erstellt einen Amazon S3 S3-Bucket, der für das Profil spezifisch

ist. Es wird verwendet, um Projektkonfigurationen zu speichern, Artefakte zu erstellen und andere Abhängigkeiten zu erstellen. Buckets werden nicht von anderen Profilen gemeinsam genutzt.

Bootstrapping beinhaltet die Schaffung neuer AWS Ressourcen und kann Kosten verursachen.

Um dein Profil zu booten:

1. Überprüfen Sie auf der Seite mit den AWS Zugangsdaten den Bootstrap-Status des Benutzerprofils, das Sie verwenden möchten. Wenn der Bootstrap-Status des Profils „Inaktiv“ lautet und kein S3-Bucket aufgeführt ist, müssen Sie das Profil booten.
2. Wählen Sie das Profil aus, das Sie verwenden möchten, und wählen Sie das Bootstrap-Profil aus.
3. Warten Sie, bis sich der Bootstrap-Status auf „Aktiv“ ändert. Dies kann einige Minuten dauern.

Plugin für Unreal: Integriere deinen Spielcode

Bevor du deinen Spieleserver in einer Flotte einsetzen kannst, musst du eine Reihe von Aktualisierungen am Spielcode vornehmen und Spielkomponenten für die Nutzung mit dem Amazon GameLift Servers Dienst zusammenstellen.

In diesem Thema werden die Schritte für eine minimale Integration beschrieben. Verwenden Sie für die Serverintegration das bereitgestellte Codebeispiel, um den Spielmodus Ihres Projekts zu aktualisieren.

- [Richten Sie Build-Ziele und Modulregeln ein](#)
- [Aktualisiere deinen Spielservercode](#)
- [Integrieren Sie Ihre Client-Game-Map](#)
- [Verpacke deine Spielkomponenten](#)

Richten Sie Build-Ziele und Modulregeln ein

Ändern Sie die Projektdateien Ihres Spiels, um die Build-Komponenten für die Verwendung mit korrekt zu generierenAmazon GameLift Servers.

So fügst du Client- und Server-Build-Ziele hinzu:

1. Öffne die Codedateien deines Spieleprojekts und suche die `.../Games/[your application name]Source/[your application name]Target.cs` Datei. Beispiel: `.../Source/`

GameLiftUnrealAppTarget.cs. (Wenn Sie Visual Studio verwenden, öffnen Sie die .sln Projektdatei.)

2. Kopieren Sie diese Datei, um zwei neue Zieldateien im Source/ Verzeichnis zu erstellen.

- Client-Ziel — Benennen Sie die neue Datei um in *[your application name]*Client.Target.cs. Bearbeiten Sie den Inhalt, um die Werte für den Klassennamen und den Zieltyp zu aktualisieren, wie im folgenden Beispielcode dargestellt:

```
using UnrealBuildTool;
using System.Collections.Generic;

public class GameLiftUnrealAppClientTarget : TargetRules
{
    public GameLiftUnrealAppClientTarget ( TargetInfo Target ) : base
    ( Target )
    {
        Type = TargetType.Client;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp" );
    }
}
```

- Serverziel — Benennen Sie die neue Datei um in *[your application name]*Server.Target.cs. Bearbeiten Sie den Inhalt, um die Werte für den Klassennamen und den Zieltyp zu aktualisieren, wie im folgenden Beispielcode dargestellt:

```
using UnrealBuildTool;
using System.Collections.Generic;

public class GameLiftUnrealAppServerTarget : TargetRules
{
    public GameLiftUnrealAppServerTarget ( TargetInfo Target ) : base
    ( Target )
    {
        Type = TargetType.Server;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp" );
    }
}
```

3. Generieren Sie Ihre Projektdateien neu. Wenn Sie Visual Studio verwenden, können Sie mit der rechten Maustaste auf die .uproject Datei Ihres Spieleprojekts klicken und Visual Studio-Projektdateien generieren auswählen.

So aktualisieren Sie die Regeln für das Spielprojektmodul:

Aktualisiere die Modulregeln des Spielprojekts, sodass sie vom Plugin abhängig sind.

1. Öffne die Codedateien deines Spielprojekts und suche die `.../Games/[your application name]Source/[your application name].Build.cs` Dateidatei. Beispiel: `.../Source/GameLiftUnrealApp.Build.cs`. (Wenn Sie Visual Studio verwenden, öffnen Sie die .sln Projektdatei.)
2. Suchen Sie die `ModuleRules` Klasse und aktualisieren Sie sie, wie im folgenden Beispielcode dargestellt:

```
using UnrealBuildTool;

public class GameLiftUnrealApp : ModuleRules
{
    public GameLiftUnrealApp ( ReadOnlyTargetRules Target ) : base ( Target )
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;
        PublicDependencyModuleNames.AddRange( new string[] { "Core",
"CoreUObject", "Engine", "InputCore", "HeadMountedDisplay",
"EnhancedInput" });
        // Add the following section
        if (Target.Type == TargetType.Server)
        {
            PublicDependencyModuleNames.Add("GameLiftServerSDK");
        }
        else
        {
            PublicDefinitions.Add("WITH_GAMELIFT=0");
        }
        bEnableExceptions = true;
    }
}
```

3. Nachdem Sie die neuen Zielformateien erstellt und die Modulregeln geändert haben, erstellen Sie Ihr Spielprojekt neu.

Aktualisiere deinen Spielservercode

Aktualisiere deinen Gameservercode, um die Kommunikation zwischen einem Gameserver-Prozess und dem Amazon GameLift Servers Dienst zu ermöglichen. Ihr Spieleserver muss in der Lage sein, auf Anfragen von zu antworten Amazon GameLift Servers, z. B. um neue Spielsitzungen zu starten und zu beenden.

Um Servercode hinzuzufügen für Amazon GameLift Servers

1. Öffnen Sie in Ihrem Code-Editor die Lösungsdatei (.sln) für Ihr Spielprojekt, die sich normalerweise im Stammordner des Projekts befindet. Beispiel: GameLiftUnrealApp.sln.
2. Suchen Sie bei geöffneter Lösung nach der Header-Datei für den Spielmodus des Projekts: [project-name]GameMode.h file. Beispiel: GameLiftUnrealAppGameMode.h.
3. Ändern Sie die Header-Datei so, dass sie dem folgenden Code entspricht. Achten Sie darauf, "GameLiftServer" durch Ihren eigenen Projektnamen zu ersetzen. Diese Updates sind spezifisch für den Spieleserver. Wir empfehlen dir, eine Sicherungskopie der ursprünglichen Spielmodusdateien zur Verwendung mit deinem Client zu erstellen.

Beispiel für einen GameMode.h-Code

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

#pragma once

#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "GameLiftUnrealAppGameMode.generated.h"

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

UCLASS(minimalapi)
class AGameLiftUnrealAppGameMode : public AGameModeBase
{
    GENERATED_BODY()

public:
    AGameLiftUnrealAppGameMode();
```

```
protected:
    virtual void BeginPlay() override;

private:
    void InitGameLift();

private:
    TSharedPtr<FProcessParameters> ProcessParameters;
};
```

- Öffnen Sie die zugehörige [project-name]GameMode.cpp Quelldatei (zum Beispiel). GameLiftUnrealAppGameMode.cpp Ändern Sie den Code so, dass er dem folgenden Beispielcode entspricht. Achten Sie darauf, "GameLiftUnrealApp" durch Ihren eigenen Projektnamen zu ersetzen. Diese Updates sind spezifisch für den Spieleserver. Wir empfehlen dir, eine Sicherungskopie der Originaldatei zur Verwendung mit deinem Client zu erstellen.

Der folgende Beispielcode zeigt, wie du die mindestens erforderlichen Elemente für die Serverintegration mit hinzufügst Amazon GameLift Servers:

- Initialisieren Sie einen Amazon GameLift Servers API-Client. Der InitSDK() Aufruf mit Serverparametern ist für eine Amazon GameLift Servers Anywhere-Flotte erforderlich. Wenn Sie eine Verbindung zu einer Anywhere-Flotte herstellen, speichert das Plugin die Serverparameter als Konsolenargumente. Der Beispielcode kann zur Laufzeit auf die Werte zugreifen.
- Implementieren Sie die erforderlichen Rückruffunktionen, um auf Anfragen des Amazon GameLift Servers Dienstes zu antworten OnStartGameSession, einschließlich OnProcessTerminate, und onHealthCheck.
- Rufen Sie ProcessReady() über einen bestimmten Port an, um den Amazon GameLift Servers Dienst zu benachrichtigen, wenn er bereit ist, Spielsitzungen abzuhalten.

Beispiel für einen Spieleservercode

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

#include "GameLiftUnrealAppGameMode.h"

#include "UObject/ConstructorHelpers.h"
```

```
#include "Kismet/GameplayStatics.h"

#if WITH_GAMELIFT
#include "GameLiftServerSDK.h"
#include "GameLiftServerSDKModels.h"
#endif

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

AGameLiftUnrealAppGameMode::AGameLiftUnrealAppGameMode() :
    ProcessParameters(nullptr)
{
    // Set default pawn class to our Blueprinted character
    static ConstructorHelpers::FClassFinder<APawn> PlayerPawnBPClass(TEXT("/Game/
ThirdPerson/Blueprints/BP_ThirdPersonCharacter"));

    if (PlayerPawnBPClass.Class != NULL)
    {
        DefaultPawnClass = PlayerPawnBPClass.Class;
    }

    UE_LOG(GameServerLog, Log, TEXT("Initializing AGameLiftUnrealAppGameMode..."));
}

void AGameLiftUnrealAppGameMode::BeginPlay()
{
    Super::BeginPlay();

#if WITH_GAMELIFT
    InitGameLift();
#endif
}

void AGameLiftUnrealAppGameMode::InitGameLift()
{
#if WITH_GAMELIFT
    UE_LOG(GameServerLog, Log, TEXT("Calling InitGameLift..."));

    // Getting the module first.
    FGameLiftServerSDKModule* GameLiftSdkModule =
    &FModuleManager::LoadModuleChecked<FGameLiftServerSDKModule>(FName("GameLiftServerSDK"));
```

```

//Define the server parameters for a GameLift Anywhere fleet. These are not needed
for a GameLift managed EC2 fleet.
FServerParameters ServerParametersForAnywhere;

bool bIsAnywhereActive = false;
if (FParse::Param(FCommandLine::Get(), TEXT("glAnywhere")))
{
    bIsAnywhereActive = true;
}

if (bIsAnywhereActive)
{
    UE_LOG(GameServerLog, Log, TEXT("Configuring server parameters for
Anywhere..."));

    // If GameLift Anywhere is enabled, parse command line arguments and pass them
in the ServerParameters object.
    FString glAnywhereWebSocketUrl = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereWebSocketUrl="),
glAnywhereWebSocketUrl))
    {
        ServerParametersForAnywhere.m_webSocketUrl =
TCHAR_TO_UTF8(*glAnywhereWebSocketUrl);
    }

    FString glAnywhereFleetId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereFleetId="),
glAnywhereFleetId))
    {
        ServerParametersForAnywhere.m_fleetId = TCHAR_TO_UTF8(*glAnywhereFleetId);
    }

    FString glAnywhereProcessId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereProcessId="),
glAnywhereProcessId))
    {
        ServerParametersForAnywhere.m_processId =
TCHAR_TO_UTF8(*glAnywhereProcessId);
    }
    else
    {
        // If no ProcessId is passed as a command line argument, generate a
randomized unique string.
        FString TimeString = FString::FromInt(std::time(nullptr));

```



```
        FString ProcessId = "ProcessId_" + TimeString;
        ServerParametersForAnywhere.m_processId = TCHAR_TO_UTF8(*ProcessId);
    }

    FString glAnywhereHostId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereHostId="),
glAnywhereHostId))
    {
        ServerParametersForAnywhere.m_hostId = TCHAR_TO_UTF8(*glAnywhereHostId);
    }

    FString glAnywhereAuthToken = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAuthToken="),
glAnywhereAuthToken))
    {
        ServerParametersForAnywhere.m_authToken =
TCHAR_TO_UTF8(*glAnywhereAuthToken);
    }

    FString glAnywhereAwsRegion = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAwsRegion="),
glAnywhereAwsRegion))
    {
        ServerParametersForAnywhere.m_awsRegion =
TCHAR_TO_UTF8(*glAnywhereAwsRegion);
    }

    FString glAnywhereAccessKey = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAccessKey="),
glAnywhereAccessKey))
    {
        ServerParametersForAnywhere.m_accessKey =
TCHAR_TO_UTF8(*glAnywhereAccessKey);
    }

    FString glAnywhereSecretKey = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereSecretKey="),
glAnywhereSecretKey))
    {
        ServerParametersForAnywhere.m_secretKey =
TCHAR_TO_UTF8(*glAnywhereSecretKey);
    }

    FString glAnywhereSessionToken = "";
```

```

        if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereSessionToken="),
glAnywhereSessionToken))
        {
            ServerParametersForAnywhere.m_sessionToken =
TCHAR_TO_UTF8(*glAnywhereSessionToken);
        }

        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_YELLOW);
        UE_LOG(GameServerLog, Log, TEXT(">>>> WebSocket URL: %s"),
*ServerParametersForAnywhere.m_webSocketUrl);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Fleet ID: %s"),
*ServerParametersForAnywhere.m_fleetId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Process ID: %s"),
*ServerParametersForAnywhere.m_processId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Host ID (Compute Name): %s"),
*ServerParametersForAnywhere.m_hostId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Auth Token: %s"),
*ServerParametersForAnywhere.m_authToken);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Aws Region: %s"),
*ServerParametersForAnywhere.m_awsRegion);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Access Key: %s"),
*ServerParametersForAnywhere.m_accessKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Secret Key: %s"),
*ServerParametersForAnywhere.m_secretKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Session Token: %s"),
*ServerParametersForAnywhere.m_sessionToken);
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }

    UE_LOG(GameServerLog, Log, TEXT("Initializing the GameLift Server..."));

    //InitSDK will establish a local connection with GameLift's agent to enable further
communication.
    FGameLiftGenericOutcome InitSdkOutcome = GameLiftSdkModule-
>InitSDK(ServerParametersForAnywhere);
    if (InitSdkOutcome.IsSuccess())
    {
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
        UE_LOG(GameServerLog, Log, TEXT("GameLift InitSDK succeeded!"));
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }
    else
    {
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
    }

```

```

    UE_LOG(GameServerLog, Log, TEXT("ERROR: InitSDK failed : ("));
    FGameLiftError GameLiftError = InitSdkOutcome.GetError();
    UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"), *GameLiftError.m_errorMessage);
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    return;
}

ProcessParameters = MakeShared<FProcessParameters>();

//When a game session is created, Amazon GameLift Servers sends an activation
request to the game server and passes along the game session object containing game
properties and other settings.
//Here is where a game server should take action based on the game session object.
//Once the game server is ready to receive incoming player connections, it should
invoke GameLiftServerAPI.ActivateGameSession()
ProcessParameters->OnStartGameSession.BindLambda([=]
(Aws::GameLift::Server::Model::GameSession InGameSession)
{
    FString GameSessionId = FString(InGameSession.GetGameSessionId());
    UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"),
*GameSessionId);
    GameLiftSdkModule->ActivateGameSession();
});

//OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback
before shutting down an instance hosting this game server.
//It gives this game server a chance to save its state, communicate with services,
etc., before being shut down.
//In this case, we simply tell Amazon GameLift Servers we are indeed going to
shutdown.
ProcessParameters->OnTerminate.BindLambda([=]()
{
    UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
    // First call ProcessEnding()
    FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule-
>ProcessEnding();
    // Then call Destroy() to free the SDK from memory
    FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();
    // Exit the process with success or failure
    if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess()) {
        UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
    }
    else {
        if (!processEndingOutcome.IsSuccess()) {

```

```

        const FGameLiftError& error = processEndingOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("ProcessEnding() failed. Error:
%s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
            *error.m_errorMessage);
    }
    if (!destroyOutcome.IsSuccess()) {
        const FGameLiftError& error = destroyOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("Destroy() failed. Error: %s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
            *error.m_errorMessage);
    }
});

//This is the HealthCheck callback.
//Amazon GameLift Servers will invoke this callback every 60 seconds or so.
//Here, a game server might want to check the health of dependencies and such.
//Simply return true if healthy, false otherwise.
//The game server has 60 seconds to respond with its health status. Amazon GameLift
Servers will default to 'false' if the game server doesn't respond in time.
//In this case, we're always healthy!
ProcessParameters->OnHealthCheck.BindLambda([]()
{
    UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));
    return true;
});

//GameServer.exe -port=7777 LOG=server.mylog
ProcessParameters->port = FURL::UrlConfig.DefaultPort;
TArray<FString> CommandLineTokens;
TArray<FString> CommandLineSwitches;

FCommandLine::Parse(FCommandLine::Get(), CommandLineTokens, CommandLineSwitches);

for (FString SwitchStr : CommandLineSwitches)
{
    FString Key;
    FString Value;

    if (SwitchStr.Split("=", &Key, &Value))
    {
        if (Key.Equals("port"))
        {

```

```

        ProcessParameters->port = FString::Atoi(*Value);
    }
}

//Here, the game server tells Amazon GameLift Servers where to find game session
log files.
//At the end of a game session, Amazon GameLift Servers uploads everything in the
specified
//location and stores it in the cloud for access later.
TArray<FString> Logfiles;
Logfiles.Add(TEXT("GameLiftUnrealApp/Saved/Logs/server.log"));
ProcessParameters->logParameters = Logfiles;

//The game server calls ProcessReady() to tell Amazon GameLift Servers it's ready
to host game sessions.
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready..."));
FGameLiftGenericOutcome ProcessReadyOutcome = GameLiftSdkModule-
>ProcessReady(*ProcessParameters);

if (ProcessReadyOutcome.IsSuccess())
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
    UE_LOG(GameServerLog, Log, TEXT("Process Ready!"));
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}
else
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
    UE_LOG(GameServerLog, Log, TEXT("ERROR: Process Ready Failed!"));
    FGameLiftError ProcessReadyError = ProcessReadyOutcome.GetError();
    UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"),
*ProcessReadyError.m_errorMessage);
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}

UE_LOG(GameServerLog, Log, TEXT("InitGameLift completed!"));
#endif
}

```

Integrieren Sie Ihre Client-Game-Map

Die Startkarte des Spiels enthält eine Entwurfslogik und Elemente der Benutzeroberfläche, die bereits Basiscode enthalten, mit dem Spielsitzungen angefordert und Verbindungsinformationen verwendet werden können, um eine Verbindung zu einer Spielsitzung herzustellen. Du kannst die Map unverändert verwenden oder sie nach Bedarf ändern. Verwenden Sie die Startkarte des Spiels zusammen mit anderen Spielressourcen, z. B. dem von Unreal Engine bereitgestellten Third-Person-Vorlagenprojekt. Diese Elemente sind im Inhaltsbrowser verfügbar. Sie können sie verwenden, um die Bereitstellungsworkflows des Plugins zu testen oder als Leitfaden für die Erstellung eines benutzerdefinierten Backend-Service für Ihr Spiel.

Die Startup-Map weist die folgenden Eigenschaften auf:

- Sie beinhaltet Logik sowohl für eine Anywhere-Flotte als auch für eine verwaltete EC2-Flotte. Wenn Sie Ihren Client ausführen, können Sie wählen, ob Sie eine Verbindung zu einer der beiden Flotten herstellen möchten.
- Zu den Client-Funktionen gehören das Suchen nach einer Spielsitzung (`SearchGameSessions()`), das Erstellen einer neuen Spielsitzung (`CreateGameSession()`) und das direkte Beitreten zu einer Spielsitzung.
- Es erhält eine eindeutige Spieler-ID aus dem Amazon Cognito Cognito-Benutzerpool Ihres Projekts (dies ist Teil einer bereitgestellten Anywhere-Lösung).

Um die Karte des Startspiels zu verwenden

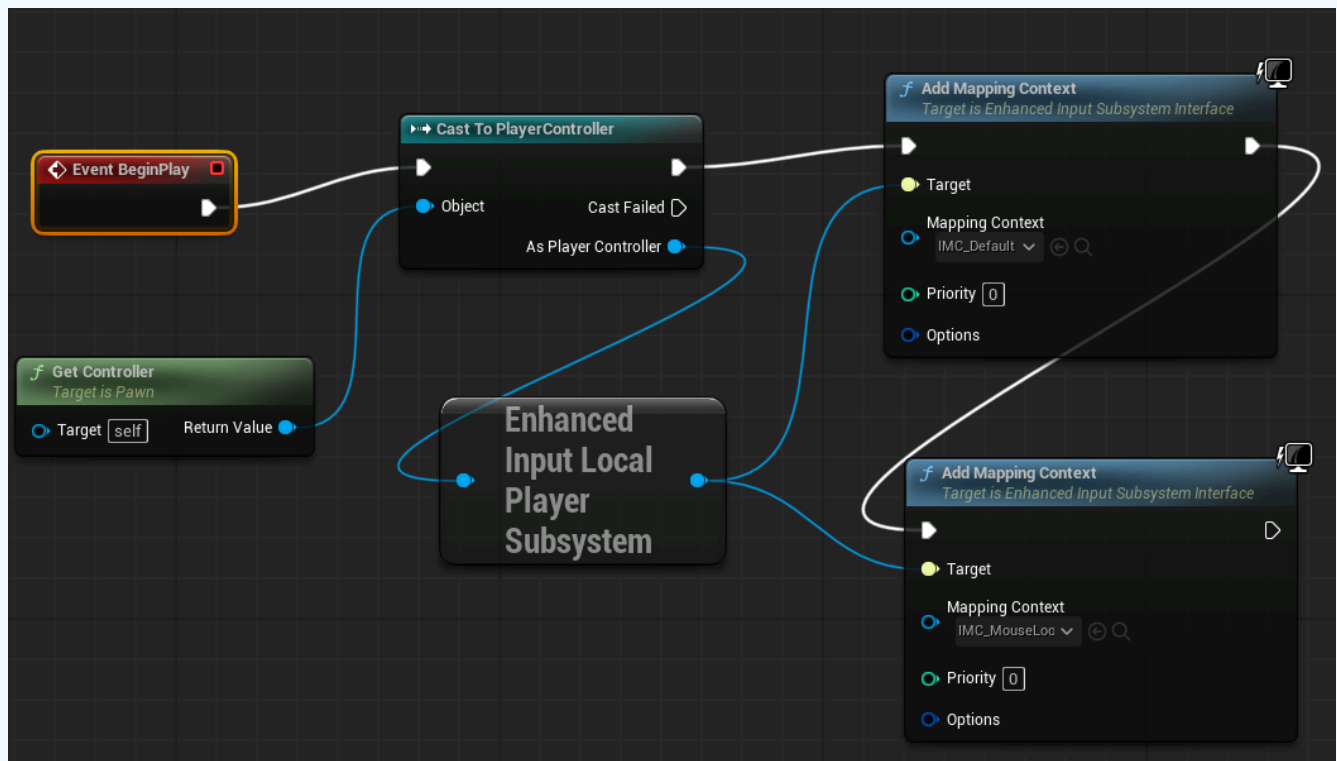
1. Öffnen Sie im UE-Editor die Seite Projekteinstellungen, Karten und Modi und erweitern Sie den Bereich Standardkarten.
2. Wählen Sie für Editor Startup Map die Option "StartupMap" aus der Dropdownliste aus. Möglicherweise müssen Sie nach der Datei suchen, die sich in `... > Unreal Projects/[project-name]/Plugins/Amazon GameLift Servers Plugin Content/Maps` befindet.
3. Wählen Sie für Game Default Map dieselbe "StartupMap" aus der Drop-down-Liste aus.
4. Wählen Sie für die Server-Standardkarte „Lv1_ThirdPerson“ für Unreal Engine 5.6 oder höher oder "ThirdPersonMap" für frühere Versionen. Dies ist eine Standardkarte, die in Ihrem Spielprojekt enthalten ist. Diese Karte ist für zwei Spieler im Spiel konzipiert.
5. Öffne das Detailfenster für die Standardkarte des Servers. Stellen Sie GameMode Override auf „Keine“ ein.

- Erweitere den Bereich Standardmodi und setze den globalen Standard-Server-Spielmodus auf den Spielmodus, den du für deine Serverintegration aktualisiert hast.

Nachdem Sie diese Änderungen an Ihrem Projekt vorgenommen haben, können Sie Ihre Spielkomponenten erstellen.

Note

Wenn du bei Unreal Engine 5.6 oder höher den Charakter nicht bewegen kannst, nachdem du eine Verbindung zum Spielserver hergestellt hast, aktualisiere den ThirdPersonCharacter BP_-Blueprint, um den Eingabe-Mapping-Kontext für IMC_Default und hinzuzufügen, IMC_MouseLook wie unten gezeigt:



Verpacke deine Spielkomponenten

Um deine Spielserver- und Spielclient-Builds zu verpacken

- Öffne dein Spielprojekt in einer Quellversion des Unreal Engine-Editors.

2. Wenn du Unreal Engine 5.6 oder höher verwendest, gehe zu Bearbeiten, Projekteinstellungen, Paketierung. Suchen Sie Cook everything im Inhaltsverzeichnis des Projekts und aktivieren Sie es.
3. Verwende den Editor, um deine Spielclient- und Server-Builds zu verpacken.
 - a. Wähle ein Ziel. Gehen Sie zu Plattformen, Windows und wählen Sie eine der folgenden Optionen aus:
 - Server: [your-application-name]Server
 - Client: [your-application-name]Client
 - b. Starte den Build. Gehen Sie zu Plattform, Windows, Paketprojekt.

Jeder Paketierungsprozess generiert eine ausführbare Datei: [your-application-name]Client.exe oder [your-application-name]Server.exe.

Legen Sie im Plugin die Pfade zu den ausführbaren Client- und Server-Build-Dateien auf Ihrer lokalen Workstation fest.

Plugin für Unreal: Hoste dein Spiel lokal mit Anywhere Amazon GameLift Servers

Verwenden Sie diesen Workflow, um Ihre lokale Workstation mithilfe einer Anywhere-Flotte als Gameserver-Host einzurichten. Du kannst ihn verwenden, um deine Gameserver-Integration zu testen, bevor du sie auf einer Cloud-basierten verwalteten Flotte installierst. Es kann auch für lokale Tests während der iterativen Spieleentwicklung nützlich sein.

Um den Amazon GameLift Servers Anywhere-Workflow zu starten:

- Wählen Sie in der Hauptwerkzeugleiste des Unreal-Editors das Amazon GameLift Servers Menü und wählen Sie Mit Anywhere hosten aus. Diese Aktion öffnet die Plugin-Seite Deploy Anywhere, auf der ein sechsstufiger Prozess zum Integrieren, Erstellen und Starten deiner Spielkomponenten vorgestellt wird.

Schritt 1: Lege dein Profil fest

Wählen Sie das Profil aus, das Sie verwenden möchten, wenn Sie diesem Workflow folgen. Das von Ihnen ausgewählte Profil wirkt sich auf alle Schritte im Workflow aus. Alle Ressourcen, die Sie erstellen, sind mit dem AWS Konto des Profils verknüpft und werden in der AWS Standardregion

des Profils platziert. Die Berechtigungen des Profilbenutzers bestimmen Ihren Zugriff auf AWS Ressourcen und Aktionen.

Um ein Benutzerprofil einzurichten

1. Wählen Sie ein Profil aus der Dropdownliste der verfügbaren Profile aus. Wenn Sie noch kein Profil haben oder ein neues erstellen möchten, gehen Sie zum `GameLiftAmazon`-Menü und wählen Sie `AWS Benutzerprofile einrichten`.
2. Wenn der Bootstrap-Status nicht „Aktiv“ lautet, wählen Sie das Bootstrap-Profil und warten Sie, bis sich der Status auf „Aktiv“ ändert.

Schritt 2: Richte deinen Spielcode ein

In diesem Schritt bereitest du deine Spieleserver- und Spielclient-Builds für die Verwendung vor Amazon GameLift Servers. Falls Sie Ihren Spielcode noch nicht integriert haben, finden Sie weitere Informationen unter [Plugin für Unreal: Integriere deinen Spielcode](#). Geben Sie die Pfade zu den ausführbaren Dateien Ihres Spiels auf Ihrer lokalen Workstation ein.

- **Spieleserver:** Integrieren Sie Ihren Spieleserver in das Server-SDK für Ihren Amazon GameLift Servers Gameserver-Build und verpacken Sie ihn. Detaillierte Anweisungen finden Sie unter [Plugin für Unreal: Integriere deinen Spielcode](#). Der Spieleserver muss in das Server-SDK integriert sein, um die Kommunikation mit dem Amazon GameLift Servers Dienst herzustellen und auf Aufforderungen zu reagieren, neue Spielsitzungen zu starten und Spielclient-Verbindungen zu akzeptieren.
- **Spielclient:** Du benötigst mindestens einen Spielclient, der mithilfe von IP-Adresse und Portinformationen eine Verbindung zu deinem Spieleserver herstellen kann. Wenn du deine Spielclient-Komponenten noch nicht eingerichtet hast, kannst du das AWS CLI Tool verwenden Amazon GameLift Servers, um manuell neue Spielsitzungen anzufordern, Verbindungsinformationen abzurufen und diese Informationen zur Verbindung mit dem Spielclient zu verwenden.

Irgendwann benötigst du einen Backend-Dienst, der Anfragen für neue Spielsitzungen an den Amazon GameLift Servers Dienst sendet und Verbindungsinformationen zurück an einen Spielclient weiterleitet. Du kannst die im Plugin enthaltenen Testmaps verwenden, um deinem Spielprojekt Amazon GameLift Servers Client-Funktionen hinzuzufügen. Hilfe zum Erstellen einer benutzerdefinierten Lösung finden Sie unter [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#).

Schritt 3: Connect zu einer Anywhere-Flotte her

In diesem Schritt bestimmen Sie eine Anywhere-Flotte, die verwendet werden soll. Eine Anywhere-Flotte definiert eine Sammlung von Rechenressourcen, die sich überall befinden können, für das Hosting von Spieleservern.

- Wenn das AWS Konto, das Sie derzeit verwenden, bereits über Anywhere-Flotten verfügt, öffnen Sie das Dropdown-Feld Flottenname und wählen Sie eine Flotte aus. In dieser Dropdownliste werden nur die Anywhere-Flotten in der AWS Region für das aktuell aktive Benutzerprofil angezeigt.
- Wenn es keine vorhandenen Flotten gibt oder Sie eine neue erstellen möchten, wählen Sie Neue Anywhere-Flotte erstellen und geben Sie einen Flottenamen ein.

Nachdem Sie eine Anywhere-Flotte für Ihr Projekt ausgewählt haben, Amazon GameLift Servers überprüft, ob der Flottenstatus aktiv ist, und zeigt die Flotten-ID an. Sie können den Fortschritt dieser Anfrage im Ausgabeprotokoll des Unreal-Editors verfolgen.

Schritt 4: Registrieren Sie Ihre Workstation

In diesem Schritt registrieren Sie Ihre lokale Workstation als Rechenressource in der neuen Anywhere-Flotte.

Um Ihre Workstation als Anywhere-Computer zu registrieren

1. Geben Sie einen Computernamen für Ihren lokalen Computer ein. Wenn Sie der Flotte mehr als einen Computer hinzufügen, müssen die Namen eindeutig sein.
2. Geben Sie eine IP-Adresse für Ihren lokalen Computer an. Dieses Feld ist standardmäßig auf die öffentliche IP-Adresse Ihres Computers eingestellt. Du kannst auch localhost (127.0.0.1) verwenden, solange du deinen Spielclient und Server auf demselben Computer ausführst.
3. Wähle Compute registrieren. Sie können den Fortschritt dieser Anfrage im Ausgabeprotokoll des Unreal-Editors verfolgen.

Amazon GameLift Servers überprüft als Antwort auf diese Aktion, ob eine Verbindung zum Computer hergestellt werden kann, und gibt Informationen über den neu registrierten Computer zurück. Außerdem werden die Konsolenargumente erstellt, die die ausführbaren Dateien Ihres Spiels benötigen, um die Kommunikation mit dem Dienst zu initialisieren. Amazon GameLift Servers

Schritt 5: Generieren Sie ein Authentifizierungstoken

Spieleserverprozesse, die auf deinem Anywhere-Computer ausgeführt werden, benötigen ein Authentifizierungstoken, um den Amazon GameLift Servers Dienst aufrufen zu können. Das Plugin generiert und speichert automatisch ein Authentifizierungstoken für die Anywhere-Flotte, wenn Sie den Spieleserver über das Plug-in starten. Der Wert des Authentifizierungstokens wird als Befehlszeilenargument gespeichert, das Ihr Servercode zur Laufzeit abrufen kann.

Die oben angegebenen Codebeispiele ermöglichen es Ihnen auch, [AWS Signature Version 4 \(Sigv4\) für API-Anfragen](#) zu verwenden. Sigv4 ist das AWS Signaturprotokoll zum Hinzufügen von Authentifizierungsinformationen zu API-Anfragen.

In diesem Schritt müssen Sie keine Maßnahmen ergreifen.

Schritt 6: Spiel starten

Zu diesem Zeitpunkt haben Sie alle Aufgaben erledigt, die zum Starten und Spielen Ihres Multiplayer-Spiels auf einer lokalen Workstation mit erforderlich sind Amazon GameLift Servers.

Um dein gehostetes Spiel zu spielen

1. Starte deinen Spieleserver. Der Spieleserver benachrichtigt Sie Amazon GameLift Servers, wenn er bereit ist, Spielsitzungen zu veranstalten.
2. Starte deinen Spielclient und verwende die neuen Funktionen, um eine neue Spielsitzung zu starten. Diese Anfrage wird Amazon GameLift Servers über den neuen Backend-Dienst an gesendet. Ruft daraufhin Amazon GameLift Servers den Spieleserver auf, der auf Ihrem lokalen Computer läuft, um eine neue Spielsitzung zu starten. Wenn die Spielsitzung bereit ist, Spieler aufzunehmen, Amazon GameLift Servers stellt es Verbindungsinformationen für den Spielclient bereit, um an der Spielsitzung teilzunehmen.

Plugin für Unreal: Stellen Sie Ihr Spiel auf einer verwalteten EC2-Flotte bereit

Stellen Sie in diesem Workflow Ihr Spiel zum Hosten auf Cloud-basierten Rechenressourcen bereit, die von verwaltet werden Amazon GameLift Servers. Laden Sie Ihren integrierten Gameserver-Build zur Bereitstellung in den Amazon GameLift Servers Dienst hoch. Falls Sie Ihren Spielcode noch nicht integriert haben, finden Sie weitere Informationen unter [Plugin für Unreal: Integriere deinen Spielcode](#). Wenn dieser Workflow abgeschlossen ist, hast du einen funktionierenden Spieleclient, der eine Verbindung zu deinen Spieleservern in der Cloud herstellen kann.

So starten Sie den Amazon GameLift Servers verwalteten Amazon EC2 EC2-Workflow:

- Wählen Sie in der Hauptsymbolleiste des Unreal-Editors das Amazon GameLift Servers Menü aus und wählen Sie Host with Managed EC2 aus. Diese Aktion öffnet die Plugin-Seite Deploy Amazon EC2 Fleet, auf der Sie in sechs Schritten Ihre Spielkomponenten integrieren, erstellen, bereitstellen und starten können.

Schritt 1: Lege dein Profil fest

Wählen Sie das Profil aus, das Sie verwenden möchten, wenn Sie diesem Workflow folgen. Das von Ihnen ausgewählte Profil wirkt sich auf alle Schritte im Workflow aus. Alle Ressourcen, die Sie erstellen, sind mit dem AWS Konto des Profils verknüpft und werden in der AWS Standardregion des Profils platziert. Die Berechtigungen des Profilbenutzers bestimmen Ihren Zugriff auf AWS Ressourcen und Aktionen.

Um ein Benutzerprofil einzurichten

1. Wählen Sie ein Profil aus der Dropdownliste der verfügbaren Profile aus. Wenn Sie noch kein Profil haben oder ein neues erstellen möchten, gehen Sie zum GameLiftAmazon-Menü und wählen Sie AWS Benutzerprofile einrichten.
2. Wenn der Bootstrap-Status nicht „Aktiv“ lautet, wählen Sie das Bootstrap-Profil und warten Sie, bis sich der Status auf „Aktiv“ ändert.

Schritt 2: Richte deinen Spielcode ein

In diesem Schritt bereitest du deine Spieleserver- und Spielclient-Builds so vor, dass sie mit dem Amazon GameLift Servers C++-Server-SDK für Unreal funktionieren. Falls du deinen Spielcode noch nicht integriert und die ausführbaren Dateien für den Spielclient und den Server noch nicht erstellt hast, findest du weitere Informationen unter. [Plugin für Unreal: Integriere deinen Spielcode](#) Geben Sie die Pfade zu den ausführbaren Dateien Ihres Spiels auf Ihrer lokalen Workstation ein.

In diesem Schritt des Workflows bietet das Plugin Links zu Anweisungen und Quellcode für die Einrichtung einer im Quellcode erstellten Version des Unreal Editors. Sie müssen die im Quellcode erstellte Version verwenden, wenn Sie Ihre Client- und Serverkomponenten erstellen.

Nachdem du einen Spieleserver erstellt hast, der in das Server-SDK integriert ist, führe die folgenden Aufgaben durch, um ihn für den Upload zum Amazon GameLift Servers Hosten vorzubereiten.

So bereitest du deinen Server-Build für die Cloud-Bereitstellung vor (Windows)

Nehmen Sie in dem `WindowsServer` Ordner, in dem der Unreal-Editor standardmäßig Ihre Server-Build-Dateien speichert, die folgenden Ergänzungen vor:

1. Kopieren Sie das Server-Build-Installationsskript in das Stammverzeichnis des **WindowsServer** Ordners. Das Installationsskript ist im Plugin-Download enthalten. Suchen Sie nach der Datei `[project-name]/Plugins/Resources/CloudFormation/extra_server_resources/install.bat`. Amazon GameLift Servers verwendet diese Datei, um den Server-Build auf Ihren Hosting-Computern zu installieren.
2. Kopieren Sie die **VC_redist.x64.exe** Datei in das Stammverzeichnis des **WindowsServer** Ordners. Sie können diesen Schritt überspringen, wenn Sie Unreal Engine Version 5.6 oder höher verwenden. Diese Datei ist in Ihrer Visual Studio-Installation enthalten. Sie befindet sich üblicherweise unter `C:/Program Files (x86)/Microsoft Visual Studio/2019/Professional/VC/Redist/MSVC/v142`.
3. Füge die OpenSSL-Bibliotheksdateien zu deinem Gameserver-Build hinzu. . Du kannst diesen Schritt überspringen, wenn dein Spieleserver in das Server-SDK 5.3 oder höher integriert ist. Diese Version ist im Amazon GameLift Servers Plugin für Unreal, Version 3.0 oder höher, enthalten.

Suchen und kopieren Sie die OpenSSL-Bibliotheken manuell in Ihr Spiele-Build-Paketverzeichnis unter `<YourGame>/Binaries/Win64`. Sie müssen dieselbe OpenSSL-Version verwenden, die Ihre Unreal Engine 5-Version verwendet. Spiele-Builds, die mit den falschen OpenSSL-Bibliotheken bereitgestellt wurden, können nicht mit dem Amazon GameLift Servers Dienst kommunizieren.

Suchen Sie in Ihrer Game-Engine-Quelle nach den OpenSSL-Bibliotheken. Der Speicherort hängt von Ihrer Entwicklungsumgebung ab:

Unter Windows:

- `[ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libimobiledevice\x64\libssl-1_1-x64.dll`
- `[ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libimobiledevice\x64\libcrypto-1_1-x64.dll`

Unter Linux:

- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libssl.so.1.1
- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libcrypto.so.1.1

Um Ihren Server-Build für die Cloud-Bereitstellung vorzubereiten (Linux)

Ausführlichere Anweisungen zur Vorbereitung eines für Linux entwickelten Spieleservers finden Sie unter [Erstellen des Server-SDK Amazon GameLift Servers für Unreal Engine 5 auf Amazon Linux](#).

1. Legen Sie ein Arbeitsverzeichnis fest, um Ihre Build-Dateien zu organisieren. Die Struktur des Arbeitsverzeichnisses wird unverändert auf jedem Hosting-Computer bereitgestellt. Füge deinen von Linux erstellten Spieleserver und alle abhängigen Dateien hinzu.
2. Erstellen Sie ein Server-Build-Installationsskript im Stammverzeichnis Ihres Arbeitsverzeichnisses. Erstellen Sie bei Bedarf eine `install.sh` Datei und fügen Sie alle Befehle hinzu, die für die korrekte Installation Ihres Gameserver-Builds erforderlich sind. Amazon GameLift Servers verwendet diese Datei, um den Server-Build auf jeder EC2-Hosting-Ressource zu installieren.
3. Füge die OpenSSL-Bibliotheksdateien zu deinem Gameserver-Build hinzu. Du kannst diesen Schritt überspringen, wenn dein Spieleserver in das Server-SDK 5.3 oder höher integriert ist.

Suchen und kopieren Sie die Bibliotheken manuell. Sie müssen dieselbe OpenSSL-Version verwenden, die Ihre Unreal Engine 5-Version verwendet. Spiele-Builds, die mit den falschen OpenSSL-Bibliotheken bereitgestellt wurden, können nicht mit dem Amazon GameLift Servers Dienst kommunizieren.

- a. Suchen Sie in Ihrer Game-Engine-Quelle nach den OpenSSL-Bibliotheken. Der Speicherort hängt von Ihrer Entwicklungsumgebung ab:

Unter Windows:

- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libmobiledevice\x64\libssl-1_1-x64.dll
- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libmobiledevice\x64\libcrypto-1_1-x64.dll

Unter Linux:

- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libssl.so.1.1

- Engine/Source/Thirdparty/openssl/1.1.1n/include/libcrypto.so.1.1
- b. Wenn du die OpenSSL-Bibliotheken gefunden hast, kopiere sie in dein Spiele-Build-Paketverzeichnis unter <YourGame>/Binaries/Linux.

Schritt 3: Wählen Sie das Bereitstellungsszenario

In diesem Schritt wählen Sie die Game-Hosting-Lösung aus, die Sie zu diesem Zeitpunkt bereitstellen möchten. Sie können Ihr Spiel in einem beliebigen Szenario mehrfach bereitstellen.

- **Flotte mit nur einer Region:** Stellt deinen Spieleserver auf einer einzigen Flotte von Hosting-Ressourcen in der Standardregion des aktiven Profils bereit. AWS Dieses Szenario ist ein guter Ausgangspunkt, um deine Serverintegration mit AWS und die Server-Build-Konfiguration zu testen. Es stellt die folgenden Ressourcen bereit:
 - AWS Fleet (On-Demand), auf dem dein Gameserver-Build installiert ist und läuft.
 - Amazon Cognito Cognito-Benutzerpool und Client, damit sich Spieler authentifizieren und ein Spiel starten können.
 - API-Gateway-Authorizer, der den Benutzerpool mit verknüpft. APIs
 - Web ACL zur Drosselung übermäßiger Spieleranrufe an das API-Gateway.
 - API-Gateway+Lambda-Funktion für Spieler, um einen Spielautomat anzufordern. Diese Funktion wird aufgerufen `CreateGameSession()`, wenn keine verfügbar sind.
 - API-Gateway+Lambda-Funktion, mit der Spieler Verbindungsinformationen für ihre Spielanfrage abrufen können.
- **FlexMatch Flotte:** Stellt deinen Spieleserver für eine Reihe von Flotten bereit und richtet einen FlexMatch Matchmaker mit Regeln für die Erstellung von Spielerspielen ein. In diesem Szenario wird kostengünstiges Spot-Hosting mit einer Struktur aus mehreren Flotten und mehreren Standorten verwendet, um eine dauerhafte Verfügbarkeit zu gewährleisten. Dieser Ansatz ist nützlich, wenn Sie bereit sind, mit der Entwicklung einer Matchmaker-Komponente für Ihre Hosting-Lösung zu beginnen. In diesem Szenario erstellen Sie die grundlegenden Ressourcen für diese Lösung, die Sie später nach Bedarf anpassen können. Sie stellt die folgenden Ressourcen bereit:
 - FlexMatch Die Matchmaking-Konfiguration und die Matchmaking-Regel ermöglichen die Annahme von Spieleranfragen und die Formierung von Matches.
 - Drei AWS Flotten, auf denen dein Gameserver-Build installiert ist und an mehreren Standorten ausgeführt wird. Beinhaltet zwei Spot-Flotten und eine On-Demand-Flotte als Backup.

- AWS Warteschlange für die Platzierung von Spielsitzungen, die Anfragen für vorgeschlagene Spiele erfüllt, indem die bestmögliche Hosting-Ressource (basierend auf Rentabilität, Kosten, Spielerlatenz usw.) gefunden und eine Spielsitzung gestartet wird.
- Amazon Cognito Cognito-Benutzerpool und Client, damit sich Spieler authentifizieren und ein Spiel starten können.
- API-Gateway-Authorizer, der den Benutzerpool mit verknüpft. APIs
- Web ACL zur Drosselung übermäßiger Spieleranrufe an das API-Gateway.
- API-Gateway+Lambda-Funktion für Spieler, um einen Spielautomat anzufordern. Diese Funktion ruft `StartMatchmaking()` auf.
- API-Gateway+Lambda-Funktion, mit der Spieler Verbindungsinformationen für ihre Spielanfrage abrufen können.
- Amazon DynamoDB-Tabellen zum Speichern von Matchmaking-Tickets für Spieler und Informationen zu Spielsitzungen.
- SNS-Thema + Lambda-Funktion zur Behandlung von Ereignissen. `GameSessionQueue`

Schritt 4: Spielparameter festlegen

In diesem Schritt beschreibst du dein Spiel zum Hochladen auf; AWS

- Name des Server-Builds: Geben Sie einen aussagekräftigen Namen für den Build Ihres Spieleservers ein. AWS verwendet diesen Namen, um auf die Kopie deines Server-Builds zu verweisen, die hochgeladen und für Bereitstellungen verwendet wurde.
- Server-Build-Betriebssystem: Geben Sie das Betriebssystem ein, auf dem Ihr Server ausgeführt werden soll. Hier erfahren Sie AWS , welche Art von Rechenressourcen für das Hosten Ihres Spiels verwendet werden sollen.
- Spieleserver-Ordner: Identifiziere den Pfad zu deinem lokalen Server-Build-Ordner.
- Spieleserver-Build: Identifiziere den Pfad zur ausführbaren Datei des Spieleservers.
- Pfad zum Spielclient: Identifiziere den Pfad zur ausführbaren Datei des Spielclients.
- Ausgabe der Client-Konfiguration: Dieses Feld muss auf einen Ordner in Ihrem Client-Build verweisen, der Ihre AWS Konfiguration enthält. Suchen Sie an der folgenden Stelle danach: `[client-build]/[project-name]/Content/CloudFormation`.

Schritt 5: Szenario bereitstellen

In diesem Schritt stellen Sie Ihr Spiel auf einer Cloud-Hosting-Lösung bereit, die auf dem von Ihnen ausgewählten Bereitstellungsszenario basiert. Dieser Vorgang kann mehrere Minuten dauern. Dabei wird Ihr Server-Build AWS validiert, Hosting-Ressourcen bereitgestellt, Ihr Spieleserver installiert, Serverprozesse gestartet und diese für das Hosten von Spielsitzungen vorbereitet.

Um mit der Bereitstellung zu beginnen, wählen Sie Deploy CloudFormation. Du kannst den Status deines Spiele-Hostings hier verfolgen. Für detailliertere Informationen können Sie sich in der AWS Management-Konsole anmelden, um Benachrichtigungen über Ereignisse einzusehen. AWS Stellen Sie sicher, dass Sie sich mit demselben Konto, demselben Benutzer und derselben AWS Region wie das aktive Benutzerprofil im Plugin anmelden.

Wenn die Bereitstellung abgeschlossen ist, haben Sie Ihren Spieleserver auf einer AWS EC2-Instance installiert. Mindestens ein Serverprozess läuft und ist bereit, eine Spielsitzung zu starten.

Schritt 6: Starten Sie den Client

Zu diesem Zeitpunkt hast du alle Aufgaben erledigt, die zum Starten und Spielen deines Multiplayer-Spiels, das mit gehostet wird, erforderlich sind Amazon GameLift Servers. Um das Spiel zu spielen, starte eine Instanz deines Spielclients.

Wenn du das Einzelflotten-Szenario eingesetzt hast, kannst du eine einzelne Client-Instanz mit einem Spieler öffnen, die Serverkarte aufrufen und dich bewegen. Öffne zusätzliche Instanzen des Spielclients, um derselben Server-Spielkarte einen zweiten Spieler hinzuzufügen.

Wenn Sie das FlexMatch Szenario bereitgestellt haben, wartet die Lösung, bis sich mindestens zwei Clients für die Platzierung der Spielsitzungen in der Warteschlange befinden, bevor die Spieler die Servermap betreten können.

Plugin für Unreal: Setze dein Spiel auf einer verwalteten Containerflotte ein

Verwende diesen geführten Plugin-Workflow, um ein Container-Image für deinen Spieleserver zu erstellen und es in einer containerbasierten Hosting-Lösung bereitzustellen. Falls du deinen Spielcode noch nicht integriert hast, findest du weitere Informationen unter. [Plugin für Unreal: Integriere deinen Spielcode](#) Wenn du diesen Workflow erfolgreich abgeschlossen hast, läuft dein containerisierter Spieleserver in der Cloud, und du kannst das Plugin verwenden, um einen Spielclient zu starten, eine Verbindung zu einer Spielsitzung herzustellen und das Spiel zu spielen.

Bevor Sie beginnen

Dieser Workflow setzt voraus, dass Sie die folgenden Aufgaben abgeschlossen haben.

- Integrieren Sie Ihren Spieleservercode in das Amazon GameLift Servers Server-SDK. Ihr gehosteter Spieleserver muss in der Lage sein, mit dem Amazon GameLift Servers Dienst zu kommunizieren, damit er auf Anfragen zum Starten neuer Spielsitzungen und zum Melden des Spielsitzungsstatus antworten kann. Wenn du diese Aufgabe noch nicht abgeschlossen hast, empfehlen wir dir, zuerst den Plugin-Workflow Host with Anywhere zu befolgen. Eine Anleitung zur Vorbereitung deines Gameservercodes findest du unter [Aktualisiere deinen Spieleservercode](#). Für eine verwaltete Containerflotte musst du dein Spiel mit dem Server-SDK Version 5.2 oder höher integrieren.

 Note

Wenn du die Startkarte des Spiels verwendest, ist diese Aufgabe bereits für dich erledigt.

- Package Sie die ausführbare Datei Ihres Gameservers für die Ausführung unter Linux. Wenn du unter Windows entwickelst und das C++-Server-SDK, Version 5.2.x oder früher, integrierst, musst du mit dem [Unreal Cross](#) Compile Toolkit arbeiten. Alternativ können Sie einen separaten Linux-Arbeitsbereich einrichten oder ein Tool wie das Windows-Subsystem für Linux (WSL) verwenden.
- Sammeln Sie Dateien, die Sie zusammen mit Ihrem Gameserver-Build bereitstellen möchten. Erstellen Sie auf Ihrem lokalen Computer ein Arbeitsverzeichnis, um die Dateien zu organisieren, die in das Container-Image Ihres Spieleservers integriert werden. Dazu können Spielabhängigkeiten, ein Skript zum Starten von Spieleservern und andere Prozesse beim Starten eines Containers usw. gehören.
- Füge die OpenSSL-Bibliotheksddateien für deinen Gameserver-Build hinzu. Du kannst diesen Schritt überspringen, wenn dein Spieleserver in das Server-SDK 5.3 oder höher integriert ist.

Suchen und kopieren Sie die OpenSSL-Bibliotheken manuell in Ihr Spiele-Build-Paketverzeichnis unter `<YourGame>/Binaries/Win64`. Sie müssen dieselbe OpenSSL-Version verwenden, die Ihre Unreal Engine 5-Version verwendet. Spiele-Builds, die mit den falschen OpenSSL-Bibliotheken bereitgestellt wurden, können nicht mit dem Amazon GameLift Servers Dienst kommunizieren.

Um die OpenSSL-Bibliotheken zu finden, schaue in deiner Game-Engine-Quelle nach. Der Speicherort hängt von Ihrer Entwicklungsumgebung ab:

Unter Windows:

- `[ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libmobiledevice\x64\libssl-1_1-x64.dll`

- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libimobiledevice\x64\libcrypto-1_1-x64.dll

Unter Linux:

- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libssl.so.1.1
- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libcrypto.so.1.1
- Integrieren Sie den Code Ihres Spielclients in Amazon GameLift Servers. Eine Möglichkeit, diese Aufgabe zu erledigen, besteht darin, ein Beispiel-Asset (im Plugin enthalten) hinzuzufügen, das bereits integriert ist. Eine Anleitung zur Vorbereitung deines Spiel-Client-Codes findest du unter [Integrieren Sie Ihre Client-Game-Map](#).
- Installieren Sie Docker auf Ihrem lokalen Computer. Sie müssen dieses Tool installiert haben, wenn Sie möchten, dass das Plugin Container-Images für Sie erstellt und in ein ECR-Repository überträgt. Alternativ kannst du diese Aufgaben manuell ausführen und das Plugin anweisen, ein vorhandenes Container-Image zu verwenden. Weitere Informationen zum manuellen Erstellen Ihres Images finden Sie unter [Erstellen eines Container-Images für Amazon GameLift Servers](#).

So starten Sie den Workflow für Amazon GameLift Servers verwaltete Container:

- Wählen Sie in der Hauptsymbolleiste des Unreal Editors das Amazon GameLift Servers Menü und dann Managed Containers aus. Diese Aktion öffnet die Plugin-Seite Host with Managed Containers, auf der step-by-step Sie ein Container-Image mit Ihrem Gameserver-Build erstellen, es auf einer Containerflotte bereitstellen und Ihr Spiel starten können.

Schritt 0: Lege dein Profil fest

In diesem Abschnitt wird Ihr aktuell ausgewähltes Benutzerprofil angezeigt. Stellen Sie sicher, dass das aktuelle Benutzerprofil das ist, das Sie für diesen Workflow verwenden möchten. Alle Ressourcen, die Sie in diesem Workflow erstellen, sind dem AWS Konto des Profils zugeordnet und werden in der AWS Standardregion des Profils platziert. Die Berechtigungen des Profilbenutzers bestimmen Ihren Zugriff auf AWS Ressourcen und Aktionen.

In folgenden Fällen müssen Sie das ausgewählte Benutzerprofil möglicherweise ändern:

- Derzeit ist kein Profil ausgewählt.
- Sie möchten ein anderes Profil auswählen oder ein neues Profil erstellen.
- Sie müssen das ausgewählte Profil booten (wenn der Bootstrap-Status inaktiv ist).

Um das ausgewählte Benutzerprofil festzulegen oder zu ändern

- Wählen Sie im Amazon GameLift Servers Menü Open AWS Access Credentials.

Schritt 1: Beurteilen Sie die Eignung von Containern

Bevor Sie Ihren Spieleserver für eine Containerflotte bereitstellen, müssen Sie ihn in ein Container-Image packen und in einem Amazon ECR-Repository speichern. Das Plugin kann diese Aufgaben für dich erledigen oder du kannst sie manuell erledigen. Geben Sie in diesem Schritt Informationen zum Status Ihres Container-Images und des ECR-Repositorys an.

Verwenden Sie die Bewertungsfragen, um dem Plugin mitzuteilen, welche Schritte es ausführen muss:

- Erstellen Sie ein neues Container-Image. Wenn du diese Option wählst, wirst du im nächsten Schritt aufgefordert, den Speicherort deines Gameserver-Build-Verzeichnisses und der ausführbaren Build-Datei anzugeben. Das Plugin verwendet eine Dockerfile-Vorlage (bereitgestellt von Amazon GameLift Servers) und konfiguriert sie automatisch für dein Spiel. Sie können die Vorlage unter einsehen. [Erstellen Sie ein Container-Image für Amazon GameLift Servers](#) Nachdem Sie diese Option ausgewählt haben, geben Sie an, wo das Plugin das neue Bild speichern soll:
 - Erstellen Sie ein neues Amazon ECR-Repository und übertragen Sie das Container-Image dorthin. Das Plugin erstellt ein privates ECR-Repo unter Verwendung des AWS Kontos und der Standardeinstellung AWS-Region in Ihrem ausgewählten Benutzerprofil.
 - Übertragen Sie das Container-Image in ein zuvor erstelltes Amazon ECR-Repository. Wenn Sie diese Option wählen, werden Sie im nächsten Schritt aufgefordert, ein vorhandenes Amazon ECR-Repository aus einer Liste auszuwählen. Die Liste enthält alle Amazon ECR-Repositorys für das AWS Konto und den Standard AWS-Region in Ihrem ausgewählten Benutzerprofil. Sie können ein öffentliches oder ein privates Repository auswählen.
- Verwenden Sie ein vorhandenes Container-Image. Wenn Sie ein Image manuell erstellt haben, empfehlen wir Ihnen, die von bereitgestellte Dockerfile-Vorlage zu verwenden Amazon GameLift Servers, die unter verfügbar ist. [Erstellen Sie ein Container-Image für Amazon GameLift Servers](#) Nachdem Sie diese Option ausgewählt haben, geben Sie an, wo sich das Bild befindet.
 - Ein lokal gespeichertes, von Docker generiertes Bild. Wenn Sie diese Option wählen, erstellt das Plugin ein neues privates Amazon ECR-Repository und überträgt die lokale Image-Datei dorthin. Im nächsten Schritt werden Sie zur Eingabe einer Bild-ID aufgefordert, die das Plugin verwendet, um die Bilddatei zu finden.

- Ein Container-Image, das bereits in einem Amazon ECR-Repository gespeichert ist. Wenn Sie diese Option wählen, werden Sie im nächsten Schritt aufgefordert, ein vorhandenes Amazon ECR-Repository und ein Bild aus einer Liste auszuwählen. Die Liste enthält alle Amazon ECR-Repositorys für das AWS Konto und den Standard AWS-Region in Ihrem ausgewählten Benutzerprofil. Sie können ein öffentliches oder ein privates Repository auswählen.

Schritt 2: Konfigurieren Sie die Image-Bereitstellung

Geben Sie in diesem Schritt die Informationen an, die das Plugin benötigt, um Ihr Container-Image für eine Containerflotte bereitzustellen. In diesem Schritt werden die folgenden Informationen angefordert:

- Der Speicherort Ihres Spieleserver-Builds, Container-Images oder Amazon ECR-Repositorys, basierend auf Ihrer Auswahl in Schritt 1.
- Das Szenario, das Sie für Ihre Bereitstellung verwalteter Container verwenden möchten.
- Der Ausgabepfad der Client-Konfiguration. Wählen Sie den Ordner in Ihrem Client-Build aus, der Ihre AWS Konfiguration enthält. Suchen Sie am folgenden Ort danach: `[client-build]/[project-name]/Content/CloudFormation`.
- Optionale Bereitstellungseinstellungen. Dieser Abschnitt enthält Konfigurationseinstellungen, die das Plugin standardmäßig verwendet. Sie können diese ändern oder die Standardwerte beibehalten
 - Der Spielname ist standardmäßig auf den Namen Ihres Spieleprojekts eingestellt. Alle AWS Ressourcen, die das Plugin erstellt, verweisen auf den Wert des Spielnamens.
 - Portbereich, Speicherlimit und vCPU-Limit sind Konfigurationseinstellungen für die Containerflotte. Weitere Informationen zum Anpassen dieser Werte finden Sie unter [Netzwerkverbindungen konfigurieren](#) Verbindungs-Portbereich und [Legen Sie Ressourcenlimits fest](#) Ressourcenlimits.
- Das Container-Image-Tag wird verwendet, um Ihre Container-Bilder in Amazon ECR zu kategorisieren. Der Standardwert ist `unreal-gamelift-plugin`.
- Name des Amazon ECR-Repositorys. Sie können dieses Feld bearbeiten, um nur dann einen benutzerdefinierten Namen vorzuschlagen, wenn das Plugin ein ECR-Repository für Sie erstellt. Der Standardwert ist `unreal-game lift-plugin-ecr-repository`.

Optionen für das Einsatzszenario

Containerflotte mit nur einer Region

In diesem Szenario wird dein Spieleserver auf einer einzigen Containerflotte bereitgestellt. Es ist ein guter Ausgangspunkt, um Ihre Serverintegration AWS und Ihre Container-Konfiguration zu testen. Es stellt die folgenden Ressourcen bereit.

- Amazon GameLift ServersDie Container-Gruppendefinition beschreibt, wie Sie Ihre Container-Images auf einer Containerflotte bereitstellen und ausführen.
- Amazon GameLift ServersContainerflotte (On-Demand) mit installiertem und laufendem Gameserver-Container mit Alias.
- Amazon Cognito Cognito-Benutzerpool und Client, damit sich Spieler authentifizieren und ein Spiel starten können.
- API Gateway Gateway-Autorisierer, der den Benutzerpool mit APIs verknüpft.
- Web Access Control List (ACL) zur Drosselung übermäßiger Player-Aufrufe an API Gateway.
- Backend-Dienst, um im Namen von Spieleclients Anfragen an den Amazon GameLift Servers Dienst zu stellen, z. B. um Spielsitzungen anzufordern und an Spielen teilzunehmen:
 - API Gateway + Lambda-Funktion für Spieler, um einen Spielsitzungsslot anzufordern. Diese Funktion wird aufgerufen `CreateGameSession()`, wenn keine offenen Slots verfügbar sind.
 - API Gateway + Lambda-Funktion, mit der Spieler Verbindungsinformationen für ihre Spielanfrage abrufen können.

Containerflotte mit einer einzigen Region FlexMatch

In diesem Szenario wird dein Spieleserver auf einer Containerflotte bereitgestellt, die Platzierung von Spielsitzungen konfiguriert und die Spielerzuweisung eingerichtet. FlexMatch Dieses Szenario ist nützlich, wenn Sie bereit sind, mit der Entwicklung eines benutzerdefinierten Matchmakers für Ihre Hosting-Lösung zu beginnen. Verwenden Sie dieses Szenario, um die grundlegenden Ressourcen für diese Lösung zu erstellen, die Sie später nach Bedarf anpassen können. Es stellt die folgenden Ressourcen bereit:

- Amazon GameLift ServersContainer-Gruppendefinition, die beschreibt, wie Sie Ihre Container-Images auf einer Containerflotte bereitstellen und ausführen.
- Amazon GameLift ServersContainerflotte (On-Demand) mit installiertem und laufendem Gameserver-Container mit Alias.

- FlexMatchKonfiguration der Spielerzuweisung und Regeln für die Spielerzuweisung zur Annahme von Spieleranfragen und zur Formierung von Matches.
- Amazon GameLift Servers Warteschlange für Spielsitzungen, in der Anfragen für vorgeschlagene Spiele bearbeitet werden, indem die bestmögliche Hosting-Ressource (basierend auf Rentabilität, Kosten, Spielerlatenz usw.) gefunden und eine Spielsitzung gestartet wird.
- Amazon Cognito Cognito-Benutzerpool und Client, damit sich Spieler authentifizieren und ein Spiel starten können.
- API Gateway Gateway-Autorisierer, der den Benutzerpool mit APIs verknüpft.
- Web Access Control List (ACL) zur Drosselung übermäßiger Player-Aufrufe an API Gateway.
- Backend-Dienst, um im Namen von Spieleclients Anfragen an den Amazon GameLift Servers Dienst zu stellen, z. B. um Spielsitzungen anzufordern und an Spielen teilzunehmen:
 - API Gateway + Lambda-Funktion für Spieler, um einen Spielsitzungsslot anzufordern. Diese Funktion wird aufgerufen `StartMatchmaking()`, wenn keine offenen Slots verfügbar sind.
 - API Gateway + Lambda-Funktion, mit der Spieler Verbindungsinformationen für ihre Spielanfrage abrufen können.
- DynamoDB-Tabellen zum Speichern von Matchmaking-Tickets für Spieler und Informationen zur Spielsitzung.
- Amazon SNS SNS-Thema + Lambda-Funktion zur Behandlung von Ereignissen.
GameSessionQueue

Setzen Sie eine Containerflotte ein

Wenn Ihre Flottenkonfiguration abgeschlossen ist, wählen Sie die Schaltfläche Containerflotte bereitstellen, um mit der Bereitstellung zu beginnen. Dieser Vorgang kann mehrere Minuten dauern, bis das Plugin ein Container-Image erstellt und an ECR weiterleitet, Hosting-Ressourcen für die Containerflotte bereitstellt, die Flotte und andere AWS Ressourcen für das ausgewählte Hosting-Lösungsszenario bereitstellt.

Wenn Sie eine Bereitstellung starten, können Sie den Fortschritt der einzelnen Schritte verfolgen. Abhängig von Ihrer Konfiguration können die Schritte Folgendes umfassen:

- Container-Image konfigurieren
- Erstellen eines Amazon ECR-Repositorys
- Ein Image erstellen und auf Amazon ECR übertragen

- Container-Gruppendefinition erstellen
- Containerflotte erstellen

Für detailliertere Informationen zur Bereitstellung wählen Sie Ansicht in der AWS Management Console. Wenn die Containerflotte den Status „Aktiv“ erreicht, laufen auf der Flotte aktiv Container mit Serverprozessen, die bereit sind, Spielsitzungen abzuhalten.

Wenn der Einsatz abgeschlossen ist, verfügen Sie über eine funktionierende Containerflotte, die bereit ist, Spielsitzungen abzuhalten und Spielerverbindungen zu akzeptieren.

Du kannst einen laufenden Einsatz nicht beenden. Wenn die Bereitstellung in einen schlechten Zustand übergeht oder fehlschlägt, können Sie mit der Option Bereitstellung zurücksetzen von vorne beginnen.

Client starten

Zu diesem Zeitpunkt hast du alle Aufgaben abgeschlossen, um dein Multiplayer-Spiel zu starten und zu spielen Amazon GameLift Servers. Um dein Spiel zu spielen, wähle Start Client, um eine lokale Instanz deines Spielclients zu starten.

- Wenn du das Einzelflotten-Szenario eingesetzt hast, öffne eine Instanz deines Spielclients mit einem Spieler und rufe die Serverkarte auf, um dich zu bewegen. Du kannst eine zweite Instanz des Spielclients öffnen, um derselben Server-Spielkarte einen zweiten Spieler hinzuzufügen.
- Wenn Sie das FlexMatch Szenario bereitgestellt haben, wartet die Hosting-Lösung darauf, dass mindestens zwei Spielclients Matchmaking-Anfragen stellen. Öffne mindestens zwei Instanzen deines Spielclients mit einem Spieler. Die beiden Spieler werden zusammengebracht und aufgefordert, an einer Spielsitzung für das Spiel teilzunehmen.

Aktualisieren Sie eine Containerflotte

Wenn Sie erfolgreich eine Hosting-Lösung für verwaltete Container bereitgestellt haben, können Sie die Funktion „Bereitstellung aktualisieren“ verwenden. Mit dieser Option können Sie die Konfigurationseinstellungen für eine bereitgestellte Containerflotte aktualisieren, ohne eine neue Flotte erstellen zu müssen.

Wenn Sie eine Bereitstellung aktualisieren, können Sie ein Container-Image mit einem anderen Gameserver-Build bereitstellen, das Amazon ECR-Repository ändern, ein anderes Bereitstellungsszenario wählen und optionale Konfigurationseinstellungen anpassen.

Wenn Sie bereit sind, Ihre Änderungen bereitzustellen, wählen Sie Aktualisieren. Die Zeit, die für ein Bereitstellungsupdate benötigt wird, ist vergleichbar mit einer vollständigen Bereitstellung. Ausführliche Informationen zur Bereitstellung erhalten Sie, wenn Sie in der AWS Management Console die Option „Ansicht“ wählen.

Bereinigen Sie die bereitgestellten Ressourcen

Es hat sich bewährt, die AWS Ressourcen für Ihre Managed-Container-Lösung zu bereinigen, sobald Sie sie nicht mehr benötigen. Möglicherweise fallen Ihnen weiterhin Kosten für diese Ressourcen an, wenn Sie sie nicht entfernen.

Löschen Sie die folgenden Ressourcen:

- **Verwalteter Container-Ressourcenstapel.** Die Ressourcen in diesem Stack hängen vom ausgewählten Bereitstellungsszenario ab. Verwenden Sie die CloudFormation Konsole, um den gesamten Stack zu löschen. Stacks, die aus dem Amazon GameLift Servers Plugin generiert werden, verwenden die folgende Namenskonvention: `GameLiftPluginForUnreal-{GameName}-Containers`. Warten Sie, bis der Vorgang zum Löschen des Stapels abgeschlossen ist, bevor Sie eine neue Bereitstellung verwalteter Container im Plugin initiieren. Weitere Informationen finden Sie unter [Löschen eines Stacks aus der CloudFormation Konsole](#).
- **Amazon ECR-Repository.** Wenn Sie das Plugin verwendet haben, um ein Repository für Ihr Container-Image zu erstellen, möchten Sie möglicherweise alle Repositories löschen, die nicht mehr benötigt werden. Sie müssen kein Repository löschen, bevor Sie eine verwaltete Container-Bereitstellung zurücksetzen. Wenn Sie ein Deployment aktualisieren oder zurücksetzen, verwendet das Plugin automatisch dasselbe Repository, sofern Sie nicht aufgefordert werden, ein anderes zu verwenden. Weitere Informationen finden Sie unter [Löschen eines privaten Repositories in Amazon ECR](#).

Amazon GameLift ServersPlugin für Unity (Server-SDK 5.x)

Dieses Plugin fügt dem Amazon GameLift Servers Unity-Editor das C#-Server-SDK und die Tools hinzu. Verwenden Sie die geführten UI-Workflows, um die Server-SDK-Funktionalität in Ihr Spielprojekt zu integrieren und eine Amazon GameLift Servers Hosting-Lösung für Ihren Spielserver bereitzustellen.

Mit dem Plugin kannst du eine grundlegende, funktionierende Hosting-Lösung erstellen und diese dann nach Bedarf optimieren und anpassen. Richten Sie eine Amazon GameLift Servers Anywhere-Flotte mit Ihrer lokalen Workstation als Host ein. Stellen Sie für Cloud-Hosting mit verwalteten EC2-

oder verwalteten Container-Flotten Ihren Spieleserver mit einer Komplettlösung zur Verwaltung von Spielesitzungsanfragen und Client-Verbindungen bereit.

Themen

- [Installieren Sie das Plugin für Ihr Unity-Spielprojekt](#)
- [Plugin für Unity: Richten Sie ein AWS Benutzerprofil ein](#)
- [Plugin für Unity: Richten Sie lokale Tests mit Amazon GameLift Servers Anywhere ein](#)
- [Plugin für Unity: Stellen Sie Ihr Spiel auf einer verwalteten EC2-Flotte bereit](#)
- [Plugin für Unity: Setze dein Spiel auf einer verwalteten Containerflotte ein](#)

Installieren Sie das Plugin für Ihr Unity-Spielprojekt

[Holen Sie sich das Amazon GameLift Servers Plugin für Unity von GitHub](#)

Informationen zur Installation des Plug-ins für ein Spielprojekt finden Sie in der Readme-Datei des GitHub Repositories.

Das Plugin beinhaltet die folgenden Komponenten:

- Plugin-Module für den Unity-Editor. Wenn das Plugin installiert ist, erhalten Sie über einen neuen Hauptmenüpunkt Zugriff auf Amazon GameLift Servers Funktionen.
- C#-Bibliotheken für die Amazon GameLift Servers Service-API mit clientseitiger Funktionalität.
- C#-Bibliotheken für das Amazon GameLift Servers Server-SDK (Version 5.x).
- Beispielhafte Spielinhalte, einschließlich Ressourcen und Szenen, sodass du sie ausprobieren kannst, Amazon GameLift Servers auch wenn du noch kein Multiplayer-Spiel hast, das fertig gebaut werden kann.
- Als CloudFormation Vorlagen bereitgestellte Lösungskonfigurationen, die das Plugin verwendet, wenn es deinen Spieleserver zum Hosten in der Cloud bereitstellt.

Dieses Plugin verwendet AWS CloudFormation Vorlagen, um Hosting-Lösungen für gängige Spieleszenarien bereitzustellen. Verwenden Sie diese Lösungen wie bereitgestellt oder passen Sie sie nach Bedarf für Ihre Spiele an.

Plugin für Unity: Richten Sie ein AWS Benutzerprofil ein

Richten Sie nach der Installation des Plugins ein Benutzerprofil mit einem gültigen AWS Konto ein. Sie können mehrere Profile im Plugin verwalten, aber Sie können jeweils nur ein Profil auswählen.

Wann immer Sie im Plugin arbeiten, wählen Sie ein Profil aus, das Sie verwenden möchten. Auf jeder Workflow-Seite wird das aktuell ausgewählte Profil angezeigt.

Wenn Sie mehrere Profile verwalten, können Sie zwischen verschiedenen Hosting-Bereitstellungen wechseln. Sie können beispielsweise Profile einrichten, die dasselbe AWS Konto verwenden, aber in verschiedenen AWS Regionen bereitgestellt werden. Oder Sie können Profile mit unterschiedlichen AWS Konten oder Benutzern und Berechtigungssätzen einrichten.

 Note

Wenn Sie die AWS CLI auf Ihrer Workstation installiert haben und bereits ein Profil konfiguriert haben, erkennt das Amazon GameLift Servers Plugin es und listet es als vorhandenes Profil auf. Das Plugin wählt automatisch jedes benannte Profil aus[default]. Sie können ein vorhandenes Profil verwenden oder ein neues erstellen.

Für alle Profile muss ein Bootstrapping durchgeführt werden, um einige erforderliche AWS Ressourcen unter Ihrem Kontonutzer einzurichten.

Um Ihre Profile zu verwalten AWS

1. Wählen Sie in der Unity-Hauptsymbolleiste das Amazon GameLift Servers Menü und wählen Sie AWS Zugangsdaten aus. Diese Aktion öffnet das Amazon GameLift Servers Plugin auf der Seite Richten Sie Ihr Benutzerprofil ein.
2. Verwenden Sie die Schaltflächen, um ein neues AWS Konto zu erstellen oder ein Benutzerprofil für ein AWS Konto einzurichten, das Sie bereits haben.
3. Wenn Sie noch kein Benutzerprofil haben, werden Sie aufgefordert, Profildetails einzugeben und ein neues Profil zu erstellen. Geben Sie die folgenden Informationen ein:
 - Ein AWS Konto. Wenn Sie ein neues AWS Konto erstellen, verwenden Sie den Link zum AWS-Managementkonsole und folgen Sie den Anweisungen. Weitere Informationen finden [Sie AWS unter Konto erstellen](#).
 - Ein AWS Benutzer mit Nutzungsberechtigungen Amazon GameLift Servers und anderen erforderlichen AWS Diensten. Anweisungen [Richten Sie ein AWS Benutzerkonto ein](#) zum Einrichten eines AWS Identity and Access Management (IAM-) Benutzers mit Amazon GameLift Servers Berechtigungen und programmatischem Zugriff mit langfristigen Anmeldeinformationen finden Sie unter.

- Anmeldeinformationen für Ihren AWS Benutzer. Diese Anmeldeinformationen bestehen aus einer AWS Zugriffsschlüssel-ID und einem AWS geheimen Schlüssel. Weitere Informationen finden Sie unter [Holen Sie sich Ihre Zugangsschlüssel](#).
 - AWS Region. Dies ist ein geografischer Standort, an dem Sie Ihre AWS Ressourcen für das Hosting erstellen möchten. Wir empfehlen, während der Entwicklung eine Region in der Nähe deines physischen Standorts zu verwenden. Wählen Sie eine Region aus der Liste der [unterstützten AWS Regionen](#) aus.
4. Wenn das Plugin vorhandene Profile erkennt, zeigt es eine Liste der verfügbaren Profile an. Wählen Sie ein vorhandenes Profil aus der Liste aus oder wählen Sie Weiteres Profil hinzufügen, um ein neues zu erstellen.

Bootstrap für ein Benutzerprofil

Alle Profile müssen bootstrapped sein, um sie mit dem Plugin verwenden zu können. Amazon GameLift Servers Bootstrapping erstellt einen Amazon S3 S3-Bucket, der für das Profil spezifisch ist. Es wird verwendet, um Projektkonfigurationen zu speichern, Artefakte zu erstellen und andere Abhängigkeiten zu erstellen. Buckets werden nicht von anderen Profilen gemeinsam genutzt.

Bootstrapping beinhaltet die Schaffung neuer AWS Ressourcen und kann Kosten verursachen.

Um dein Profil zu booten:

1. Überprüfen Sie auf der Seite mit den AWS Zugangsdaten den Bootstrap-Status des Benutzerprofils, das Sie verwenden möchten. Wenn der Bootstrap-Status des Profils „Inaktiv“ lautet und kein S3-Bucket aufgeführt ist, müssen Sie das Profil booten.
2. Wählen Sie das Profil aus, das Sie verwenden möchten, und wählen Sie das Bootstrap-Profil aus.
3. Warten Sie, bis sich der Bootstrap-Status auf „Aktiv“ ändert. Dies kann einige Minuten dauern.

Plugin für Unity: Richten Sie lokale Tests mit Amazon GameLift Servers Anywhere ein

In diesem Workflow fügen Sie Client- und Server-Spielcode für die Amazon GameLift Servers Funktionalität hinzu und verwenden das Plug-in, um Ihre lokale Workstation als Testspielserver-Host festzulegen. Wenn du die Integrationsaufgaben abgeschlossen hast, verwende das Plugin, um deinen Spielclient und deine Serverkomponenten zu erstellen.

Um den Amazon GameLift Servers Anywhere-Workflow zu starten:

- Wählen Amazon GameLift Servers und wählen Sie im Hauptmenü des Unity-Editors Host with Anywhere aus. Diese Aktion öffnet die Plugin-Seite, auf der du dein Spiel mit einer @Anywhere - Flotte einrichten kannst. Auf dieser Seite wird ein fünfstufiger Prozess zur Integration, Erstellung und Veröffentlichung deiner Spielkomponenten vorgestellt.

Lege dein Profil fest

Wählen Sie das Profil aus, das Sie verwenden möchten, wenn Sie diesem Workflow folgen. Das von Ihnen ausgewählte Profil wirkt sich auf alle Schritte im Workflow aus. Alle Ressourcen, die Sie erstellen, sind mit dem AWS Konto des Profils verknüpft und werden in der AWS Standardregion des Profils platziert. Die Berechtigungen des Profilbenutzers bestimmen Ihren Zugriff auf AWS Ressourcen und Aktionen.

1. Wählen Sie ein Profil aus der Dropdownliste der verfügbaren Profile aus. Wenn Sie noch kein Profil haben oder ein neues erstellen möchten, gehen Sie zum Amazon GameLift Servers Menü und wählen Sie AWS Kontoprofile einrichten.
2. Wenn der Bootstrap-Status nicht „Aktiv“ lautet, wählen Sie das Bootstrap-Profil und warten Sie, bis sich der Status auf „Aktiv“ ändert.

Integrieren Sie Ihren Spielcode in das C#-Server-SDK

Note

Wenn du das Beispielspiel importiert hast, kannst du diesen Schritt überspringen. In den Beispielspiel-Assets ist bereits der erforderliche Server- und Client-Code vorhanden.

Für diesen Schritt im Workflow aktualisierst du den Client- und Servercode in deinem Spielprojekt.

- Spieleserver müssen in der Lage sein, mit dem Amazon GameLift Servers Dienst zu kommunizieren, um Aufforderungen zum Starten einer Spielsitzung zu erhalten, Verbindungsinformationen zur Spielsitzung bereitzustellen und den Status zu melden.
- Spieleclients müssen in der Lage sein, Informationen über Spielsitzungen abzurufen, an Spielsitzungen teilzunehmen oder sie zu starten und Verbindungsinformationen abzurufen, um an einem Spiel teilzunehmen.

Integrieren Sie Ihren Servercode

Wenn du dein eigenes Spielprojekt mit benutzerdefinierten Szenen verwendest, verwende den bereitgestellten Beispielcode, um deinem Spielprojekt den erforderlichen Servercode hinzuzufügen:

1. Öffnen Sie den `Assets/Scripts/Server` Ordner in Ihren Spielprojektdateien. Wenn er nicht existiert, erstelle ihn.
2. Gehe zum GitHub Repo [aws/ amazon-gamelift-plugin-unity](https://github.com/aws/amazon-gamelift-plugin-unity) und öffne den Pfad. `Samples~/SampleGame/Assets/Scripts/Server`
3. Suchen Sie die Datei `GameLiftServer.cs` und kopieren Sie sie in den Ordner Ihres Spielprojekts. Server Wenn Sie eine ausführbare Serverdatei erstellen, verwenden Sie diese Datei als Build-Ziel.

Der Beispielcode enthält die folgenden mindestens erforderlichen Elemente, die das Amazon GameLift Servers C#-Server-SDK (Version 5) verwenden:

- Initialisiert einen Amazon GameLift Servers API-Client. Der `InitSDK()` Aufruf mit Serverparametern ist für eine Amazon GameLift Servers Anywhere-Flotte erforderlich. Diese Einstellungen werden automatisch für die Verwendung im Plugin festgelegt.
- Implementiert die erforderlichen Rückruffunktionen, um auf Anfragen des Amazon GameLift Servers Dienstes zu antworten, einschließlich `OnStartGameSessionOnProcessTerminate`, `undonHealthCheck`.
- Ruft `ProcessReady()` über einen bestimmten Port auf, um den Amazon GameLift Servers Dienst zu benachrichtigen, wenn der Serverprozess bereit ist, Spielsitzungen zu hosten.

Wenn Sie den Beispielservercode anpassen möchten, sehen Sie sich diese Ressourcen an:

- [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#)
- [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)

Integrieren Sie Ihren Client-Code

Wenn du dein eigenes Spielprojekt mit benutzerdefinierten Szenen verwendest, musst du grundlegende Funktionen in deinen Spielclient integrieren. Außerdem musst du UI-Elemente hinzufügen, damit sich Spieler anmelden und an einer Spielsitzung teilnehmen können. Verwenden Sie die Service-API für Amazon GameLift Servers (im AWS SDK), um Informationen zu

Spielsitzungen abzurufen, neue Spielsitzungen zu erstellen oder bestehenden Spielsitzungen beizutreten.

Wenn Sie einen Client für lokale Tests mit einer Anywhere-Flotte erstellen, können Sie dem Amazon GameLift Servers Service direkte Aufrufe hinzufügen. Wenn Sie Ihr Spiel für Cloud-Hosting entwickeln — oder wenn Sie planen, Anywhere-Flotten für das Produktions-Hosting zu verwenden — müssen Sie einen clientseitigen Backend-Dienst einrichten, der die gesamte Kommunikation zwischen Spieleclients und dem Dienst abwickelt. Amazon GameLift Servers

Verwenden Sie die folgenden Ressourcen als Leitfaden für Amazon GameLift Servers die Integration in Ihren Client-Code.

- Integrieren Sie den Client mit der `GameLiftCoreApi` Klasse im GitHub Repo `amazon-gamelift-plugin-unity aws/`. Diese Klasse bietet Steuerelemente für die Spielerauthentifizierung und das Abrufen von Informationen zur Spielsitzung.
- Sehen Sie sich Beispiele für Spielintegrationen an, die im GitHub Repo `aws/`, verfügbar sind. `amazon-gamelift-plugin-unity Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`
- Folgen Sie den Anweisungen unter `Zu Ihrem Amazon GameLift Servers Unity-Spieleclient hinzufügen`.

Für Spieleclients, die eine Verbindung zu einer Anywhere-Flotte herstellen, benötigt Ihr Spieleclient die folgenden Informationen. Das Plugin aktualisiert dein Spielprojekt automatisch, sodass es die Ressourcen verwendet, die du im Plugin erstellt hast.

- `FleetId` - Die eindeutige Kennung für Ihre Anywhere-Flotte.
- `FleetLocation` - Der benutzerdefinierte Standort Ihrer Anywhere-Flotte.
- `AwsRegion` - Die AWS Region, in der Ihre Anywhere-Flotte gehostet wird. Dies ist die Region, die Sie in Ihrem Benutzerprofil festgelegt haben.
- `ProfileName` - Ein AWS Anmeldeinformationsprofil auf Ihrem lokalen Computer, das den Zugriff auf das AWS SDK für ermöglicht Amazon GameLift Servers. Der Spielclient verwendet diese Anmeldeinformationen, um Anfragen an den Amazon GameLift Servers Dienst zu authentifizieren.

 Note

Das Profil mit den Anmeldeinformationen wird vom Plugin generiert und auf dem lokalen Computer gespeichert. Daher müssen Sie den Client auf dem lokalen Computer (oder auf einem Computer mit demselben Profil) ausführen.

Stellen Sie eine Connect zu einer Anywhere-Flotte her

In diesem Schritt bestimmen Sie eine Anywhere-Flotte, die verwendet werden soll. Eine Anywhere-Flotte definiert eine Sammlung von Rechenressourcen, die sich überall befinden können, für das Hosting von Spieleservern.

- Wenn das AWS Konto, das Sie derzeit verwenden, bereits über Anywhere-Flotten verfügt, öffnen Sie das Dropdown-Feld Flottenname und wählen Sie eine Flotte aus. In dieser Dropdownliste werden nur die Anywhere-Flotten in der AWS Region für das aktuell aktive Benutzerprofil angezeigt.
- Wenn es keine vorhandenen Flotten gibt oder Sie eine neue erstellen möchten, wählen Sie Neue Anywhere-Flotte erstellen und geben Sie einen Flottenamen ein.

Nachdem Sie eine Anywhere-Flotte für Ihr Projekt ausgewählt haben, Amazon GameLift Servers überprüft, ob der Flottenstatus aktiv ist, und zeigt die Flotten-ID an. Sie können den Fortschritt dieser Anfrage im Ausgabeprotokoll des Unity-Editors verfolgen.

Registrieren Sie einen Computer

In diesem Schritt registrieren Sie Ihre lokale Workstation als Rechenressource in der neuen Anywhere-Flotte.

1. Geben Sie einen Rechenamen für Ihren lokalen Computer ein. Wenn Sie der Flotte mehr als einen Computer hinzufügen, müssen die Namen eindeutig sein.
2. Wählen Sie Compute registrieren aus. Sie können den Fortschritt dieser Anfrage im Ausgabeprotokoll des Unity-Editors verfolgen.

Das Plugin registriert Ihre lokale Workstation mit der IP-Adresse, die auf localhost (127.0.0.1) eingestellt ist. Bei dieser Einstellung wird davon ausgegangen, dass du deinen Spielclient und Server auf demselben Computer ausführst.

Amazon GameLift Servers überprüft als Reaktion auf diese Aktion, ob eine Verbindung zum Computer hergestellt werden kann, und gibt Informationen über den neu registrierten Computer zurück.

Spiel starten

In diesem Schritt baust du deine Spielkomponenten und startest sie, um das Spiel zu spielen. Führen Sie die folgenden Schritte aus:

1. Konfiguriere deinen Spielclient. In diesem Schritt forderst du das Plugin auf, ein `GameLiftClientSettings` Asset für dein Spielprojekt zu aktualisieren. Das Plugin verwendet dieses Asset, um bestimmte Informationen zu speichern, die dein Spielclient benötigt, um eine Verbindung zum Amazon GameLift Servers Dienst herzustellen.
 - a. Wenn du das Beispielspiel nicht importiert und initialisiert hast, erstelle ein neues `GameLiftClientSettings` Asset. Wählen Sie im Hauptmenü des Unity-Editors die Optionen Assets, Create, Amazon GameLift, Client Settings aus. Wenn Sie `GameLiftClientSettings` in Ihrem Projekt mehrere Kopien von erstellen, erkennt das Plugin dies automatisch und benachrichtigt Sie, welches Asset das Plugin aktualisieren wird.
 - b. Wähle unter Spiel starten die Option Client konfigurieren: Anywhere-Einstellungen anwenden. Diese Aktion aktualisiert deine Spielclient-Einstellungen, sodass du die Anywhere-Flotte verwendest, die du gerade eingerichtet hast.
2. Erstelle deinen Spielclient und starte ihn.
 - a. Erstellen Sie eine ausführbare Client-Datei mithilfe des Standard-Unity-Build-Prozesses. Wechseln Sie unter Datei, Build-Einstellungen die Plattform zu Windows, Mac, Linux. Wenn du das Beispielspiel importiert und die Einstellungen initialisiert hast, werden die Build-Liste und das Build-Ziel automatisch aktualisiert.
 - b. Startet eine oder mehrere Instanzen der neu erstellten ausführbaren Datei für den Spielclient.
3. Starte einen Gameserver in deiner Anywhere-Flotte. Wählen Sie Server: Server im Editor starten. Diese Aufgabe startet einen Live-Server, zu dem Ihr Client eine Verbindung herstellen kann, solange der Unity-Editor geöffnet bleibt.
4. Starte eine Spielsitzung oder nimm an einer Spielsitzung teil. Verwenden Sie in Ihren Spielclient-Instanzen die Benutzeroberfläche, um jeden Client zu einer Spielsitzung hinzuzufügen. Wie du das machst, hängt davon ab, wie du dem Client Funktionen hinzugefügt hast.

Wenn du den Beispiel-Spielclient verwendest, weist er die folgenden Eigenschaften auf:

- Eine Komponente zur Spielanmeldung. Wenn eine Verbindung zu einem Spieleserver auf einer Anywhere-Flotte hergestellt wird, erfolgt keine Spielvalidierung. Sie können beliebige Werte eingeben, um an der Spielsitzung teilzunehmen.
- Eine einfache Benutzeroberfläche für die Teilnahme am Spiel. Wenn ein Client versucht, einem Spiel beizutreten, sucht der Client automatisch nach einer aktiven Spielsitzung mit einem verfügbaren Spielerplatz. Wenn keine Spielsitzung verfügbar ist, fordert der Client eine neue Spielsitzung an. Wenn eine Spielsitzung verfügbar ist, fordert der Client die Teilnahme an der verfügbaren Spielsitzung an. Wenn du dein Spiel mit mehreren gleichzeitigen Clients testest, startet der erste Client die Spielsitzung, und die übrigen Clients nehmen automatisch an der bestehenden Spielsitzung teil.
- Spielsitzungen mit Slots für vier Spieler. Du kannst bis zu vier Spielclient-Instanzen gleichzeitig starten und sie nehmen an derselben Spielsitzung teil.

Von einer ausführbaren Serverdatei aus starten (optional)

Sie können die ausführbare Datei Ihres Spieleservers erstellen und starten, um sie auf einer Anywhere-Flotte zu testen.

1. Erstellen Sie eine ausführbare Serverdatei mithilfe des standardmäßigen Unity-Build-Prozesses. Stellen Sie unter Datei, Build-Einstellungen die Plattform auf Dedicated Server um und erstellen Sie.
2. Holen Sie sich ein kurzfristiges Authentifizierungstoken, indem Sie den AWS CLI-Befehl [get-compute-auth-token](#) mit Ihrer Anywhere-Flotten-ID und AWS Region aufrufen. Die Flotten-ID wird in Connect to an Anywhere Fleet angezeigt, wenn Sie die Flotte erstellen. Die AWS Region wird unter „Profil einrichten“ angezeigt, wenn Sie Ihr aktives Profil auswählen.

```
aws gamelift get-compute-auth-token --fleet-id [your anywhere fleet ID] --region [your AWS region]
```

3. Starte die neu erstellte ausführbare Spielserver-Datei über eine Befehlszeile und übergebe ein gültiges Authentifizierungstoken.

```
my_project.exe --authToken [token]
```

Plugin für Unity: Stellen Sie Ihr Spiel auf einer verwalteten EC2-Flotte bereit

In diesem Workflow verwenden Sie das Plugin, um Ihr Spiel für das Hosten auf Cloud-basierten Rechenressourcen vorzubereiten, die von Amazon GameLift Servers verwaltet werden. Sie fügen Client- und Server-Spielcode für die Amazon GameLift Servers Funktionalität hinzu und laden dann Ihren Server-Build zum Hosten in den Amazon GameLift Servers Dienst hoch. Wenn dieser Workflow abgeschlossen ist, laufen deine Spieleserver in der Cloud und ein funktionierender Spieleclient, der eine Verbindung zu ihnen herstellen kann.

So starten Sie den Amazon GameLift Servers verwalteten Amazon EC2 EC2-Workflow:

- Wählen Amazon GameLift Servers und wählen Sie im Hauptmenü des Unity-Editors Host with Managed EC2 aus. Dieser Workflow umfasst einen sechsstufigen Prozess zum Integrieren, Erstellen, Bereitstellen und Starten Ihrer Spielkomponenten.

Lege dein Profil fest

Wählen Sie das Profil aus, das Sie verwenden möchten, wenn Sie diesem Workflow folgen. Das von Ihnen ausgewählte Profil wirkt sich auf alle Schritte im Workflow aus. Alle Ressourcen, die Sie erstellen, sind mit dem AWS Konto des Profils verknüpft und werden in der AWS Standardregion des Profils platziert. Die Berechtigungen des Profilbenutzers bestimmen Ihren Zugriff auf AWS Ressourcen und Aktionen.

1. Wählen Sie ein Profil aus der Dropdownliste der verfügbaren Profile aus. Wenn Sie noch kein Profil haben oder ein neues erstellen möchten, gehen Sie zum Amazon GameLift Servers Menü und wählen Sie AWS Kontoprofile einrichten.
2. Wenn der Bootstrap-Status nicht „Aktiv“ lautet, wählen Sie das Bootstrap-Profil und warten Sie, bis sich der Status auf „Aktiv“ ändert.

Integriere dein Spiel mit Amazon GameLift Servers

Für diese Aufgabe aktualisierst du den Client- und Servercode in deinem Spielprojekt.

- Spieleserver müssen in der Lage sein, mit dem Amazon GameLift Servers Dienst zu kommunizieren, um Aufforderungen zum Starten einer Spielsitzung zu erhalten, Verbindungsinformationen zur Spielsitzung bereitzustellen und den Status zu melden.

- Spieleclients müssen in der Lage sein, Informationen über Spielsitzungen abzurufen, an Spielsitzungen teilzunehmen oder sie zu starten und Verbindungsinformationen abzurufen, um an einem Spiel teilzunehmen.

Note

Wenn du das Beispielspiel importiert hast, kannst du diesen Schritt überspringen. In den Beispielspiel-Assets ist bereits der erforderliche Server- und Client-Code vorhanden.

Integrieren Sie Ihren Servercode

Wenn Sie Ihr eigenes Spieleprojekt mit benutzerdefinierten Szenen verwenden, verwenden Sie den bereitgestellten Beispielcode, um Ihrem Spielprojekt den erforderlichen Servercode hinzuzufügen. Wenn Sie Ihr Spielprojekt zu Testzwecken in eine Anywhere-Flotte integriert haben, haben Sie die Anweisungen in diesem Schritt bereits ausgeführt.

1. Öffnen Sie in den Projektdateien Ihres Spiels den Assets/Scripts/Server Ordner. Wenn er nicht existiert, erstelle ihn.
2. Gehe zum GitHub Repo [aws/ amazon-gamelift-plugin-unity](#) und öffne den Pfad. Samples~/SampleGame/Assets/Scripts/Server
3. Suchen Sie die Datei `GameLiftServer.cs` und kopieren Sie sie in den Ordner Ihres Spielprojekts. Server Wenn Sie eine ausführbare Serverdatei erstellen, verwenden Sie diese Datei als Build-Ziel.

Der Beispielcode enthält die folgenden mindestens erforderlichen Elemente, die das Amazon GameLift Servers C#-Server-SDK (Version 5) verwenden:

- Initialisiert einen Amazon GameLift Servers API-Client. Der `initSDK()`-Aufruf mit Serverparametern ist für eine Amazon GameLift Servers Anywhere-Flotte erforderlich. Diese Einstellungen werden automatisch für die Verwendung im Plugin festgelegt.
- Implementiert die erforderlichen Rückruffunktionen, um auf Anfragen des Amazon GameLift Servers Dienstes zu antworten, einschließlich `OnStartGameSessionOnProcessTerminate`, `undonHealthCheck`.
- Ruft `ProcessReady()` über einen bestimmten Port auf, um den Amazon GameLift Servers Dienst zu benachrichtigen, wenn der Serverprozess bereit ist, Spielsitzungen zu hosten.

Wenn Sie den Beispielservercode anpassen möchten, sehen Sie sich diese Ressourcen an:

- [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#)
- [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)

Integrieren Sie Ihren Client-Code

Für Spieleclients, die eine Verbindung zu cloudbasierten Spieleservern herstellen, empfiehlt es sich, einen clientseitigen Backend-Dienst zu verwenden, um den Amazon GameLift Servers Dienst aufzurufen, anstatt die Aufrufe direkt vom Spielclient aus zu tätigen.

Im Plugin-Workflow für das Hosting auf einer verwalteten EC2-Flotte umfasst jedes Bereitstellungsszenario einen vorgefertigten Backend-Service, der die folgenden Komponenten umfasst:

- Eine Reihe von Lambda-Funktionen und DynamoDB-Tabellen, die zum Anfordern von Spielsitzungen und zum Abrufen von Spielsitzungsinformationen verwendet werden. Diese Komponenten verwenden ein API-Gateway als Proxy.
- Ein Amazon Cognito Cognito-Benutzerpool, der eindeutige Spieler generiert IDs und Spielerverbindungen authentifiziert.

Um diese Komponenten verwenden zu können, benötigt Ihr Spielclient die Möglichkeit, Anfragen an den Backend-Service zu senden, um Folgendes zu tun:

- Erstellen Sie einen Player-Benutzer im AWS Cognito-Benutzerpool und authentifizieren Sie sich.
- Nehmen Sie an einer Spielsitzung teil und erhalten Sie Verbindungsinformationen.
- Nehmen Sie über Matchmaking an einem Spiel teil.

Verwenden Sie die folgenden Ressourcen als Leitfaden.

- Integrieren Sie den Client mit der [GameLiftCoreApi](#) Klasse im GitHub Repo [amazon-gamelift-plugin-unityaws/](#). Diese Klasse bietet Steuerelemente für die Spielerauthentifizierung und das Abrufen von Informationen zur Spielsitzung.
- [Die Beispiel-Integrationen für Spiele finden Sie im GitHub Repo aws/.. amazon-gamelift-plugin-unity Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs](#)
- [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität.](#)

(Nur Unity 6.3+) Integrieren Sie das Skript `install.sh`

Unity 6.3 unter Linux benötigt Glibc 2.35. Amazon Linux 2023 enthält Glibc 2.34. Es wird ein Skript `install.sh` bereitgestellt, das das Erstellen von Glibc 2.35 aus dem Quellcode und die Konfiguration der Unity-Binärdateien für die Verwendung von Glibc 2.35 übernimmt.

1. [Gehe zum Repo aws/ und öffne den Pfad GitHub . amazon-gamelift-plugin-unity](#)
`ExtraServerResources`
2. Suchen Sie die Datei `install.sh` und kopieren Sie sie in den Ordner, der Ihren Server-Build enthält.
3. Wenn Sie über ein vorhandenes Skript verfügen `install.sh`, führen Sie den Inhalt beider Skripts zu einem einzigen `zusammeninstall.sh`.

Wählen Sie ein Einsatzszenario

In diesem Schritt wählen Sie die Game-Hosting-Lösung aus, die Sie zu diesem Zeitpunkt bereitstellen möchten. Sie können Ihr Spiel in einem beliebigen Szenario mehrfach bereitstellen.

- Flotte mit nur einer Region: Stellt deinen Spielservers auf einer einzigen Flotte von Hosting-Ressourcen in der Standardregion des aktiven Profils bereit. AWS Dieses Szenario ist ein guter Ausgangspunkt, um deine Serverintegration mit AWS und die Server-Build-Konfiguration zu testen. Es stellt die folgenden Ressourcen bereit:
 - AWS Fleet (On-Demand), auf dem dein Gameserver-Build installiert ist und läuft.
 - Amazon Cognito Cognito-Benutzerpool und Client, damit sich Spieler authentifizieren und ein Spiel starten können.
 - API-Gateway-Authorizer, der den Benutzerpool mit verknüpft. APIs
 - Web ACL zur Drosselung übermäßiger Spieleranrufe an das API-Gateway.
 - API-Gateway+Lambda-Funktion für Spieler, um einen Spielautomat anzufordern. Diese Funktion wird aufgerufen `CreateGameSession()`, wenn keine verfügbar sind.
 - API-Gateway+Lambda-Funktion, mit der Spieler Verbindungsinformationen für ihre Spielanfrage abrufen können.
- FlexMatch Flotte: Stellt deinen Spielservers für eine Reihe von Flotten bereit und richtet einen FlexMatch Matchmaker mit Regeln für die Erstellung von Spielerspielen ein. In diesem Szenario wird kostengünstiges Spot-Hosting mit einer Struktur aus mehreren Flotten und mehreren Standorten verwendet, um eine dauerhafte Verfügbarkeit zu gewährleisten. Dieser Ansatz ist nützlich, wenn Sie bereit sind, mit der Entwicklung einer Matchmaker-Komponente für Ihre Hosting-

Lösung zu beginnen. In diesem Szenario erstellen Sie die grundlegenden Ressourcen für diese Lösung, die Sie später nach Bedarf anpassen können. Sie stellt die folgenden Ressourcen bereit:

- FlexMatch Die Matchmaking-Konfiguration und die Matchmaking-Regel ermöglichen die Annahme von Spieleranfragen und die Formierung von Matches.
- Drei AWS Flotten, auf denen dein Gameserver-Build installiert ist und an mehreren Standorten ausgeführt wird. Beinhaltet zwei Spot-Flotten und eine On-Demand-Flotte als Backup.
- AWS Warteschlange für die Platzierung von Spielsitzungen, die Anfragen für vorgeschlagene Spiele erfüllt, indem die bestmögliche Hosting-Ressource (basierend auf Rentabilität, Kosten, Spielerlatenz usw.) gefunden und eine Spielsitzung gestartet wird.
- Amazon Cognito Cognito-Benutzerpool und Client, damit sich Spieler authentifizieren und ein Spiel starten können.
- API-Gateway-Authorizer, der den Benutzerpool mit verknüpft. APIs
- Web ACI zur Drosselung übermäßiger Spieleranrufe an das API-Gateway.
- API-Gateway+Lambda-Funktion für Spieler, um einen Spielautomat anzufordern. Diese Funktion ruft `StartMatchmaking()` auf.
- API-Gateway+Lambda-Funktion, mit der Spieler Verbindungsinformationen für ihre Spielanfrage abrufen können.
- Amazon DynamoDB-Tabellen zum Speichern von Matchmaking-Tickets für Spieler und Informationen zu Spielsitzungen.
- SNS-Thema + Lambda-Funktion zur Behandlung von Ereignissen. `GameSessionQueue`

Stellen Sie die Spielparameter ein

In diesem Schritt beschreibst du dein Spiel, auf das du hochladen möchtest AWS.

- Spielname: Gib einen aussagekräftigen Namen für dein Spielprojekt ein. Dieser Name wird innerhalb des Plugins verwendet.
- Flottenname: Geben Sie einen aussagekräftigen Namen für Ihre verwaltete EC2-Flotte ein. Amazon GameLift Servers verwendet diesen Namen (zusammen mit der Flotten-ID), wenn Ressourcen in der AWS Konsole aufgelistet werden.
- Build-Name: Geben Sie einen aussagekräftigen Namen für Ihren Server-Build ein. AWS verwendet diesen Namen, um auf die Kopie Ihres Server-Builds zu verweisen, die in Bereitstellungen hochgeladen Amazon GameLift Servers und dort verwendet wurde.

- **Startparameter:** Geben Sie optionale Anweisungen ein, die beim Starten der ausführbaren Serverdatei auf einer verwalteten EC2-Flotteninstanz ausgeführt werden sollen. Die maximale Länge beträgt 1024 Zeichen.
- **Spieler-Ordner:** Geben Sie den Pfad zu einem lokalen Ordner an, der Ihren Server-Build enthält.
- **Spieler-Datei:** Geben Sie den Namen der ausführbaren Serverdatei an.

Szenario bereitstellen

In diesem Schritt stellen Sie Ihr Spiel auf der Grundlage des von Ihnen ausgewählten Bereitstellungsszenarios in einer Cloud-Hosting-Lösung bereit. Dieser Vorgang kann mehrere Minuten dauern. Dabei wird Ihr Serverbuild AWS validiert, Hosting-Ressourcen bereitgestellt, Ihr Spieler installiert, Serverprozesse gestartet und diese für das Hosten von Spielsitzungen vorbereitet.

Um mit der Bereitstellung zu beginnen, wählen Sie **Deploy CloudFormation**. Du kannst den Status deines Spiele-Hostings hier verfolgen. Für detailliertere Informationen können Sie sich in der AWS Management-Konsole anmelden, um Benachrichtigungen über Ereignisse einzusehen. AWS Stellen Sie sicher, dass Sie sich mit demselben Konto, demselben Benutzer und derselben AWS Region wie das aktive Benutzerprofil im Plugin anmelden.

Wenn die Bereitstellung abgeschlossen ist, haben Sie Ihren Spieler auf einer AWS EC2-Instance installiert. Mindestens ein Serverprozess läuft und ist bereit, eine Spielsitzung zu starten.

Starte den Spielclient

Wenn deine Flotte erfolgreich eingesetzt wurde, laufen deine Spieler jetzt und stehen für Spielsitzungen zur Verfügung. Du kannst jetzt deinen Client erstellen, ihn starten und eine Verbindung herstellen, um an der Spielsitzung teilzunehmen.

1. Konfiguriere deinen Spielclient. In diesem Schritt forderst du das Plugin auf, ein **GameLiftClientSettings** Asset für dein Spielprojekt zu aktualisieren. Das Plugin verwendet dieses Asset, um bestimmte Informationen zu speichern, die dein Spielclient benötigt, um eine Verbindung zum Amazon GameLift Servers Dienst herzustellen.
 - a. Wenn du das Beispielspiel nicht importiert und initialisiert hast, erstelle ein neues **GameLiftClientSettings** Asset. Wählen Sie im Hauptmenü des Unity-Editors die Optionen **Assets, Create, Amazon GameLift, Client Settings** aus. Wenn Sie

GameLiftClientSettings in Ihrem Projekt mehrere Kopien von erstellen, erkennt das Plugin dies automatisch und benachrichtigt Sie, welches Asset das Plugin aktualisieren wird.

- b. Wähle unter Spiel starten die Option Client konfigurieren: Verwaltete EC2-Einstellungen anwenden. Diese Aktion aktualisiert deine Spielclient-Einstellungen, sodass sie die verwaltete EC2-Flotte verwenden, die du gerade bereitgestellt hast.
2. Erstelle deinen Spielclient. Erstellen Sie eine ausführbare Client-Datei mithilfe des Standard-Unity-Build-Prozesses. Wechseln Sie unter Datei, Build-Einstellungen die Plattform zu Windows, Mac, Linux. Wenn du das Beispielspiel importiert und die Einstellungen initialisiert hast, werden die Build-Liste und das Build-Ziel automatisch aktualisiert.
3. Startet die neu erstellte ausführbare Datei für den Spielclient. Um mit dem Spielen des Spiels zu beginnen, starte zwei bis vier Client-Instanzen und verwende jeweils die Benutzeroberfläche, um an einer Spielsitzung teilzunehmen.

Wenn du den Beispiel-Spielclient verwendest, weist er die folgenden Eigenschaften auf:

- Eine Komponente zur Spielanmeldung. Wenn eine Verbindung zu einem Spieleserver auf einer Anywhere-Flotte hergestellt wird, erfolgt keine Spielvalidierung. Sie können beliebige Werte eingeben, um an der Spielsitzung teilzunehmen.
- Eine einfache Benutzeroberfläche für die Teilnahme am Spiel. Wenn ein Client versucht, einem Spiel beizutreten, sucht der Client automatisch nach einer aktiven Spielsitzung mit einem verfügbaren Spielerplatz. Wenn keine Spielsitzung verfügbar ist, fordert der Client eine neue Spielsitzung an. Wenn eine Spielsitzung verfügbar ist, fordert der Client die Teilnahme an der verfügbaren Spielsitzung an. Wenn du dein Spiel mit mehreren gleichzeitigen Clients testest, startet der erste Client die Spielsitzung, und die übrigen Clients nehmen automatisch an der bestehenden Spielsitzung teil.
- Spielsitzungen mit Slots für vier Spieler. Du kannst bis zu vier Spielclient-Instanzen gleichzeitig starten und sie nehmen an derselben Spielsitzung teil.

Plugin für Unity: Setze dein Spiel auf einer verwalteten Containerflotte ein

Verwende diesen geführten Plugin-Workflow, um ein Container-Image für deinen Spieleserver zu erstellen und es in einer containerbasierten Hosting-Lösung bereitzustellen. Wenn du diesen Workflow erfolgreich abgeschlossen hast, läuft dein containerisierter Spieleserver in der Cloud und du kannst das Plugin verwenden, um einen Spielclient zu starten, eine Verbindung zu einer Spielsitzung herzustellen und das Spiel zu spielen.

Bevor Sie beginnen

Dieser Workflow setzt voraus, dass Sie die folgenden Aufgaben abgeschlossen haben.

- Integrieren Sie Ihren Spieleservercode in das Amazon GameLift Servers Server-SDK. Unser gehosteter Spieleserver muss in der Lage sein, mit dem Amazon GameLift Servers Dienst zu kommunizieren, damit er auf Anfragen zum Starten neuer Spielsitzungen und zum Melden des Spielsitzungsstatus reagieren kann. Wenn du diese Aufgabe noch nicht abgeschlossen hast, empfehlen wir dir, zuerst den Plugin-Workflow Host with Anywhere zu befolgen. Eine Anleitung zur Vorbereitung deines Gameservercodes findest du unter [Integrieren Sie Ihren Servercode](#). Für eine verwaltete Containerflotte musst du dein Spiel mit dem Server-SDK Version 5.2 oder höher integrieren.

Note

Wenn du das Beispielspiel importiert hast, ist diese Aufgabe bereits für dich erledigt.

- Package Sie die ausführbare Datei Ihres Gameservers für die Ausführung unter Linux.
- Sammeln Sie Dateien, die Sie zusammen mit Ihrem Gameserver-Build bereitstellen möchten. Erstellen Sie auf Ihrem lokalen Computer ein Arbeitsverzeichnis, um die Dateien zu organisieren, die in das Container-Image Ihres Spieleservers integriert werden. Dazu können Spielabhängigkeiten, ein Skript zum Starten von Spieleservern und andere Prozesse beim Starten eines Containers usw. gehören.
- Integrieren Sie Ihren Spielclient-Code in Amazon GameLift Servers. Eine Möglichkeit, diese Aufgabe zu erledigen, besteht darin, ein Beispiel-Asset (im Plugin enthalten) hinzuzufügen, das bereits integriert ist. Eine Anleitung zur Vorbereitung deines Spiel-Client-Codes findest du unter [Integrieren Sie Ihren Client-Code](#).
- Installieren Sie Docker auf Ihrem lokalen Computer. Sie müssen dieses Tool installiert haben, wenn Sie möchten, dass das Plugin Container-Images für Sie erstellt und in ein ECR-Repository überträgt. Alternativ kannst du diese Aufgaben manuell ausführen und das Plugin anweisen, ein vorhandenes Container-Image zu verwenden. Weitere Informationen zum manuellen Erstellen Ihres Images finden Sie unter [Erstellen eines Container-Images für Amazon GameLift Servers](#).

So starten Sie den Workflow für Amazon GameLift Servers verwaltete Container:

- Wählen Sie in der Hauptsymbolleiste des Unity-Editors das Amazon GameLift Servers Menü und dann Managed Containers aus. Diese Aktion öffnet die Plugin-Seite Host with Managed

Containers, auf der step-by-step Sie ein Container-Image mit Ihrem Gameserver-Build erstellen, es auf einer Containerflotte bereitstellen und Ihr Spiel starten können.

Schritt 0: Lege dein Profil fest

In diesem Abschnitt wird Ihr aktuell ausgewähltes Benutzerprofil angezeigt. Stellen Sie sicher, dass das aktuelle Benutzerprofil das ist, das Sie für diesen Workflow verwenden möchten. Alle Ressourcen, die Sie in diesem Workflow erstellen, sind dem AWS Konto des Profils zugeordnet und werden in der AWS Standardregion des Profils platziert. Die Berechtigungen des Profilbenutzers bestimmen Ihren Zugriff auf AWS Ressourcen und Aktionen.

In folgenden Fällen müssen Sie das ausgewählte Benutzerprofil möglicherweise ändern:

- Derzeit ist kein Profil ausgewählt.
- Sie möchten ein anderes Profil auswählen oder ein neues Profil erstellen.
- Sie müssen das ausgewählte Profil booten (wenn der Bootstrap-Status inaktiv ist).

Um das ausgewählte Benutzerprofil festzulegen oder zu ändern

- Wählen Sie im Amazon GameLift Servers Menü Open AWS Access Credentials.

Schritt 1: Beurteilen Sie die Eignung von Containern

Bevor Sie Ihren Spieleserver für eine Containerflotte bereitstellen, müssen Sie ihn in ein Container-Image packen und in einem Amazon ECR-Repository speichern. Das Plugin kann diese Aufgaben für dich erledigen oder du kannst sie manuell erledigen. Geben Sie in diesem Schritt Informationen zum Status Ihres Container-Images und des ECR-Repositorys an.

Verwenden Sie die Bewertungsfragen, um dem Plugin mitzuteilen, welche Schritte es ausführen muss:

- Erstellen Sie ein neues Container-Image. Wenn du diese Option wählst, wirst du im nächsten Schritt aufgefordert, den Speicherort deines Gameserver-Build-Verzeichnisses und der ausführbaren Build-Datei anzugeben. Das Plugin verwendet eine Dockerfile-Vorlage (bereitgestellt von Amazon GameLift Servers) und konfiguriert sie automatisch für dein Spiel. Sie können die Vorlage unter einsehen. [Erstellen Sie ein Container-Image für Amazon GameLift Servers](#) Nachdem Sie diese Option ausgewählt haben, geben Sie an, wo das Plugin das neue Bild speichern soll:

- Erstellen Sie ein neues Amazon ECR-Repository und übertragen Sie das Container-Image dorthin. Das Plugin erstellt ein privates ECR-Repo unter Verwendung des AWS Kontos und der Standardeinstellung AWS-Region in Ihrem ausgewählten Benutzerprofil.
- Übertragen Sie das Container-Image in ein zuvor erstelltes Amazon ECR-Repository. Wenn Sie diese Option wählen, werden Sie im nächsten Schritt aufgefordert, ein vorhandenes Amazon ECR-Repository aus einer Liste auszuwählen. Die Liste enthält alle Amazon ECR-Repositorys für das AWS Konto und den Standard AWS-Region in Ihrem ausgewählten Benutzerprofil. Sie können ein öffentliches oder ein privates Repository auswählen.
- Verwenden Sie ein vorhandenes Container-Image. Wenn Sie ein Image manuell erstellt haben, empfehlen wir Ihnen, die von bereitgestellte Dockerfile-Vorlage zu verwendenAmazon GameLift Servers, die unter verfügbar ist. [Erstellen Sie ein Container-Image für Amazon GameLift Servers](#) Nachdem Sie diese Option ausgewählt haben, geben Sie an, wo sich das Bild befindet.
 - Ein lokal gespeichertes, von Docker generiertes Bild. Wenn Sie diese Option wählen, erstellt das Plugin ein neues privates Amazon ECR-Repository und überträgt die lokale Image-Datei dorthin. Im nächsten Schritt werden Sie zur Eingabe einer Bild-ID aufgefordert, die das Plugin verwendet, um die Bilddatei zu finden.
 - Ein Container-Image, das bereits in einem Amazon ECR-Repository gespeichert ist. Wenn Sie diese Option wählen, werden Sie im nächsten Schritt aufgefordert, ein vorhandenes Amazon ECR-Repository und ein Bild aus einer Liste auszuwählen. Die Liste enthält alle Amazon ECR-Repositorys für das AWS Konto und den Standard AWS-Region in Ihrem ausgewählten Benutzerprofil. Sie können ein öffentliches oder ein privates Repository auswählen.
- Unity 6.3 unter Linux benötigt Glibc 2.35. Amazon Linux 2023 enthält Glibc 2.34. Im Folgenden finden Sie eine Dockerfile-Vorlage, die das Erstellen von Glibc 2.35 aus dem Quellcode und die Konfiguration der Unity-Binärdateien für die Verwendung von Glibc 2.35 übernimmt.

Dockerfile-Vorlage für ein Container-Image für einen Unity 6.3-Spieleserver

Diese Vorlage enthält die Mindestanweisungen, die ein Spieleserver-Container benötigt, um in einer Amazon GameLift Servers Flotte verwendet zu werden. Ändern Sie den Inhalt nach Bedarf für Ihren Spieleserver.

```
# Base image
# -----
# Add the base image that you want to use over here,
# Make sure to use an image with the same architecture as the
# Instance type you are planning to use on your fleets.
```

```
FROM public.ecr.aws/amazonlinux/amazonlinux

# Game build directory
# -----
# Add your game build directory in the 'GAME_BUILD_DIRECTORY' env variable below.
#
# Game executable
# -----
# Add the relative path to your executable in the 'GAME_EXECUTABLE' env variable
below.
# The game build provided over here needs to be integrated with gamelift server sdk.
# This template assumes that the executable path is relative to the game build
directory.
#
# Launch parameters
# -----
# Add any launch parameters to pass into your executable in the 'LAUNCH_PARAMS' env
variable below.
#
# Default directory
# -----
# The value provided in 'HOME_DIR' below will be where the game executable and logs
exist.
#
ARG GAME_BUILD_DIRECTORY
ARG GAME_EXECUTABLE
ARG LAUNCH_PARAMS

ENV GAME_BUILD_DIRECTORY=$GAME_BUILD_DIRECTORY \
    GAME_EXECUTABLE=$GAME_EXECUTABLE \
    LAUNCH_PARAMS=$LAUNCH_PARAMS \
    HOME_DIR="/local/game"

# install dependencies as necessary
RUN yum install -y shadow-utils bison wget gcc make patchelf tar gzip && \
    yum clean all && rm -fr /var/cache

RUN mkdir -p $HOME_DIR
COPY .$GAME_BUILD_DIRECTORY/ $HOME_DIR

# Change directory to home
WORKDIR $HOME_DIR

# Build glibc 2.35 and patch Unity binaries
```

```

RUN GLIBC_VERSION="2.35" && \
  INSTALL_PREFIX="/opt/glibc-${GLIBC_VERSION}" && \
  ARCH=$(uname -m) && \
  if [ "$ARCH" = "x86_64" ]; then \
    INTERPRETER="${INSTALL_PREFIX}/lib/ld-linux-x86-64.so.2"; \
    MONO_ARCH="x86_64"; \
  elif [ "$ARCH" = "aarch64" ]; then \
    INTERPRETER="${INSTALL_PREFIX}/lib/ld-linux-aarch64.so.1"; \
    MONO_ARCH="aarch64"; \
  else \
    echo "ERROR: Unsupported architecture: $ARCH"; exit 1; \
  fi && \
  cd /tmp && \
  wget -q "https://ftp.gnu.org/gnu/glibc/glibc-${GLIBC_VERSION}.tar.gz" && \
  tar -xzf "glibc-${GLIBC_VERSION}.tar.gz" && \
  cd "glibc-${GLIBC_VERSION}" && mkdir glibc-build && cd glibc-build && \
  touch /etc/ld.so.conf && \
  ../configure --prefix="${INSTALL_PREFIX}" && \
  make -s all && make -s install && \
  cd / && rm -rf /tmp/glibc-* && \
  GLIBC_LIB="${INSTALL_PREFIX}/lib" && \
  EXECUTABLE_NAME=$(echo "$GAME_EXECUTABLE" | sed 's/\.[^\.]*$//') && \
  DATA_DIR="$HOME_DIR/${EXECUTABLE_NAME}_Data" && \
  patchelf --set-interpreter "$INTERPRETER" "$HOME_DIR/$GAME_EXECUTABLE" && \
  patchelf --set-rpath "$GLIBC_LIB:$HOME_DIR:/lib64:$DATA_DIR/MonoBleedingEdge/
$MONO_ARCH" "$HOME_DIR/$GAME_EXECUTABLE" && \
  patchelf --set-rpath "$GLIBC_LIB:/lib64" "$HOME_DIR/UnityPlayer.so" && \
  MONO_LIB="$DATA_DIR/MonoBleedingEdge/$MONO_ARCH/libmonobdwgc-2.0.so" && \
  if [ -f "$MONO_LIB" ]; then patchelf --set-rpath "$GLIBC_LIB:/lib64" "$MONO_LIB";
fi && \
  GAME_ASSEMBLY="$HOME_DIR/GameAssembly.so" && \
  if [ -f "$GAME_ASSEMBLY" ]; then patchelf --set-rpath "$GLIBC_LIB:/lib64"
"$GAME_ASSEMBLY"; fi

RUN useradd -m gamescale && \
  echo "gamescale ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers && \
  chown -R gamescale:gamescale $HOME_DIR

# Add permissions to game build
RUN chmod +x ./${GAME_EXECUTABLE}

USER gamescale

# Check directory before starting the container

```

```
RUN ls -lhrt .

# Check path before starting the container
RUN echo $PATH

# Start the game build
ENTRYPOINT ["/bin/sh", "-c", "./$GAME_EXECUTABLE $LAUNCH_PARAMS"]
```

Schritt 2: Konfigurieren Sie die Image-Bereitstellung

Geben Sie in diesem Schritt die Informationen an, die das Plugin benötigt, um Ihr Container-Image für eine Containerflotte bereitzustellen. In diesem Schritt werden die folgenden Informationen angefordert:

- Der Speicherort Ihres Spieleserver-Builds, Container-Images oder Amazon ECR-Repositorys, basierend auf Ihrer Auswahl in Schritt 1.
- Das Szenario, das Sie für Ihre Bereitstellung verwalteter Container verwenden möchten.
- Optionale Bereitstellungseinstellungen. Dieser Abschnitt enthält Konfigurationseinstellungen, die das Plugin standardmäßig verwendet. Sie können diese ändern oder die Standardwerte beibehalten
 - Der Spielname ist standardmäßig auf den Namen Ihres Spieleprojekts eingestellt. Alle AWS Ressourcen, die das Plugin erstellt, verweisen auf den Wert des Spielnamens.
 - Portbereich, Speicherlimit und vCPU-Limit sind Konfigurationseinstellungen für die Containerflotte. Weitere Informationen zum Anpassen dieser Werte finden Sie unter [Netzwerkverbindungen konfigurieren](#) Verbindungs-Portbereich und [Legen Sie Ressourcenlimits fest](#) Ressourcenlimits.
 - Das Container-Image-Tag wird verwendet, um Ihre Container-Bilder in Amazon ECR zu kategorisieren. Der Standardwert ist `unity-gamelift-plugin`.

Optionen für das Einsatzszenario

Containerflotte mit nur einer Region

In diesem Szenario wird dein Spieleserver auf einer einzigen Containerflotte bereitgestellt. Es ist ein guter Ausgangspunkt, um Ihre Serverintegration AWS und Ihre Container-Konfiguration zu testen. Es stellt die folgenden Ressourcen bereit.

- Amazon GameLift ServersDie Container-Gruppengröße beschreibt, wie Sie Ihre Container-Images auf einer Containerflotte bereitstellen und ausführen.
- Amazon GameLift ServersContainerflotte (On-Demand) mit installiertem und laufendem Gameserver-Container mit Alias.
- Amazon Cognito Cognito-Benutzerpool und Client, damit sich Spieler authentifizieren und ein Spiel starten können.
- API Gateway Gateway-Autorisierer, der den Benutzerpool mit APIs verknüpft.
- Web Access Control List (ACL) zur Drosselung übermäßiger Player-Aufrufe an API Gateway.
- Backend-Dienst, um im Namen von Spieleclients Anfragen an den Amazon GameLift Servers Dienst zu stellen, z. B. um Spielsitzungen anzufordern und an Spielen teilzunehmen:
 - API Gateway + Lambda-Funktion für Spieler, um einen Spielsitzungsslot anzufordern. Diese Funktion wird aufgerufen `CreateGameSession()`, wenn keine offenen Slots verfügbar sind.
 - API Gateway + Lambda-Funktion, mit der Spieler Verbindungsinformationen für ihre Spielanfrage abrufen können.

Containerflotte mit einer Region FlexMatch

In diesem Szenario wird dein Spieleserver auf einer Containerflotte bereitgestellt, die Platzierung von Spielsitzungen konfiguriert und die Spielerzuweisung eingerichtet. FlexMatch Dieses Szenario ist nützlich, wenn Sie bereit sind, mit der Entwicklung eines benutzerdefinierten Matchmakers für Ihre Hosting-Lösung zu beginnen. Verwenden Sie dieses Szenario, um die grundlegenden Ressourcen für diese Lösung zu erstellen, die Sie später nach Bedarf anpassen können. Es stellt die folgenden Ressourcen bereit:

- Amazon GameLift ServersContainer-Gruppengröße, die beschreibt, wie Sie Ihre Container-Images auf einer Containerflotte bereitstellen und ausführen.
- Amazon GameLift ServersContainerflotte (On-Demand) mit installiertem und laufendem Gameserver-Container mit Alias.
- FlexMatchKonfiguration der Spielerzuweisung und Regeln für die Spielerzuweisung zur Annahme von Spieleranfragen und zur Formierung von Matches.
- Amazon GameLift ServersWarteschlange für Spielsitzungen, in der Anfragen für vorgeschlagene Spiele bearbeitet werden, indem die bestmögliche Hosting-Ressource (basierend auf Rentabilität, Kosten, Spielerlatenz usw.) gefunden und eine Spielsitzung gestartet wird.
- Amazon Cognito Cognito-Benutzerpool und Client, damit sich Spieler authentifizieren und ein Spiel starten können.

- API Gateway Gateway-Autorisierer, der den Benutzerpool mit APIs verknüpft.
- Web Access Control List (ACL) zur Drosselung übermäßiger Player-Aufrufe an API Gateway.
- Backend-Dienst, um im Namen von Spieleclients Anfragen an den Amazon GameLift Servers Dienst zu stellen, z. B. um Spielsitzungen anzufordern und an Spielen teilzunehmen:
 - API Gateway + Lambda-Funktion für Spieler, um einen Spielsitzungsslot anzufordern. Diese Funktion wird aufgerufen `StartMatchmaking()`, wenn keine offenen Slots verfügbar sind.
 - API Gateway + Lambda-Funktion, mit der Spieler Verbindungsinformationen für ihre Spielanfrage abrufen können.
- DynamoDB-Tabellen zum Speichern von Matchmaking-Tickets für Spieler und Informationen zur Spielsitzung.
- Amazon SNS SNS-Thema + Lambda-Funktion zur Behandlung von Ereignissen.
GameSessionQueue

Setzen Sie eine Containerflotte ein

Wenn Ihre Flottenkonfiguration abgeschlossen ist, wählen Sie die Schaltfläche Containerflotte bereitstellen, um mit der Bereitstellung zu beginnen. Dieser Vorgang kann mehrere Minuten dauern, bis das Plugin ein Container-Image erstellt und an ECR weiterleitet, Hosting-Ressourcen für die Containerflotte bereitstellt, die Flotte und andere AWS Ressourcen für das ausgewählte Hosting-Lösungsszenario bereitstellt.

Wenn Sie eine Bereitstellung starten, können Sie den Fortschritt der einzelnen Schritte verfolgen. Abhängig von Ihrer Konfiguration können die Schritte Folgendes umfassen:

- Container-Image konfigurieren
- Erstellen eines Amazon ECR-Repositorys
- Ein Image erstellen und auf Amazon ECR übertragen
- Container-Gruppendefinition erstellen
- Containerflotte erstellen

Für detailliertere Informationen zur Bereitstellung wählen Sie Ansicht in der AWS Management Console. Wenn die Containerflotte den Status „Aktiv“ erreicht, laufen auf der Flotte aktiv Container mit Serverprozessen, die bereit sind, Spielsitzungen abzuhalten.

Wenn der Einsatz abgeschlossen ist, verfügen Sie über eine funktionierende Containerflotte, die bereit ist, Spielsitzungen abzuhalten und Spielerverbindungen zu akzeptieren.

Du kannst einen laufenden Einsatz nicht beenden. Wenn die Bereitstellung in einen schlechten Zustand übergeht oder fehlschlägt, können Sie mit der Option Bereitstellung zurücksetzen von vorne beginnen.

Client starten

Zu diesem Zeitpunkt hast du alle Aufgaben abgeschlossen, um dein Multiplayer-Spiel zu starten und zu spielen Amazon GameLift Servers. Um dein Spiel zu spielen, wähle Start Client, um eine lokale Instanz deines Spielclients zu starten.

- Wenn du das Einzelflotten-Szenario eingesetzt hast, öffne eine Instanz deines Spielclients mit einem Spieler und rufe die Serverkarte auf, um dich zu bewegen. Du kannst eine zweite Instanz des Spielclients öffnen, um derselben Server-Spielkarte einen zweiten Spieler hinzuzufügen.
- Wenn Sie das FlexMatch Szenario bereitgestellt haben, wartet die Hosting-Lösung darauf, dass mindestens zwei Spielclients Matchmaking-Anfragen stellen. Öffne mindestens zwei Instanzen deines Spielclients mit einem Spieler. Die beiden Spieler werden zusammengebracht und aufgefordert, an einer Spielsitzung für das Spiel teilzunehmen.

Aktualisieren Sie eine Containerflotte

Wenn Sie erfolgreich eine Hosting-Lösung für verwaltete Container bereitgestellt haben, können Sie die Funktion „Bereitstellung aktualisieren“ verwenden. Mit dieser Option können Sie die Konfigurationseinstellungen für eine bereitgestellte Containerflotte aktualisieren, ohne eine neue Flotte erstellen zu müssen.

Wenn Sie eine Bereitstellung aktualisieren, können Sie ein Container-Image mit einem anderen Gameserver-Build bereitstellen, das Amazon ECR-Repository ändern, ein anderes Bereitstellungsszenario wählen und optionale Konfigurationseinstellungen anpassen.

Wenn Sie bereit sind, Ihre Änderungen bereitzustellen, wählen Sie Aktualisieren. Die Zeit, die für ein Bereitstellungsupdate benötigt wird, ist vergleichbar mit einer vollständigen Bereitstellung. Ausführliche Informationen zur Bereitstellung erhalten Sie, wenn Sie in der AWS Management Console die Option „Ansicht“ wählen.

Bereinigen Sie die bereitgestellten Ressourcen

Es hat sich bewährt, die AWS Ressourcen für Ihre Managed-Container-Lösung zu bereinigen, sobald Sie sie nicht mehr benötigen. Möglicherweise fallen Ihnen weiterhin Kosten für diese Ressourcen an, wenn Sie sie nicht entfernen.

Löschen Sie die folgenden Ressourcen:

- Stapel verwalteter Container-Ressourcen. Die Ressourcen in diesem Stack hängen vom ausgewählten Bereitstellungsszenario ab. Verwenden Sie die CloudFormation Konsole, um den gesamten Stack zu löschen. Stacks, die aus dem Amazon GameLift Servers Plugin generiert werden, verwenden die folgende Namenskonvention: `GameLiftPluginForUnity-{GameName}-Containers`. Warten Sie, bis der Vorgang zum Löschen des Stapels abgeschlossen ist, bevor Sie eine neue Bereitstellung verwalteter Container im Plugin initiieren. Weitere Informationen finden Sie unter [Löschen eines Stacks aus der CloudFormation Konsole](#).
- Amazon ECR-Repository. Wenn Sie das Plugin verwendet haben, um ein Repository für Ihr Container-Image zu erstellen, möchten Sie möglicherweise alle Repositories löschen, die nicht mehr benötigt werden. Sie müssen kein Repository löschen, bevor Sie eine verwaltete Container-Bereitstellung zurücksetzen. Wenn Sie ein Deployment aktualisieren oder zurücksetzen, verwendet das Plugin automatisch dasselbe Repository, sofern Sie nicht aufgefordert werden, ein anderes zu verwenden. Weitere Informationen finden Sie unter [Löschen eines privaten Repositories in Amazon ECR](#).

Amazon GameLift ServersPlugin für Unity für Server SDK 4

Note

Dieses Thema enthält Informationen zu einer früheren Version des Amazon GameLift Servers Plugins für Unity. Version 1.x verwendet das Server-SDK für Amazon GameLift Servers 4.x oder früher. Dokumentation zur neuesten Plugin-Version, die das Server-SDK 5.x verwendet und neuere Funktionen wie Amazon GameLift Servers Anywhere und verwaltetes Container-Hosting unterstützt, finden Sie unter [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#)

Amazon GameLift Servers stellt Tools zur Verfügung, mit denen Sie Ihre Multiplayer-Spieleserver für die Ausführung vorbereiten können. Amazon GameLift Servers Das Amazon GameLift Servers Plugin für Unity erleichtert die Integration Amazon GameLift Servers in Ihre Unity-Spieleprojekte und die Bereitstellung von Amazon GameLift Servers Ressourcen für das Cloud-Hosting. Verwenden Sie das Plugin für Unity, um auf AWS CloudFormation Vorlagen für gängige Spieleszenarien zuzugreifen Amazon GameLift Servers APIs und diese bereitzustellen.

Nachdem Sie das Plugin eingerichtet haben, können Sie das [Amazon GameLift ServersUnity-Beispiel](#) ausprobieren GitHub.

Themen

- [Amazon GameLift ServersIntegrieren Sie in ein Unity-Gameserver-Projekt](#)
- [Amazon GameLift ServersIntegrieren Sie in ein Unity-Spieleclient-Projekt](#)
- [Installieren und richten Sie das Plugin ein](#)
- [Teste dein Spiel lokal](#)
- [Stellen Sie ein Szenario bereit](#)
- [Integrieren Sie Spiele Amazon GameLift Servers in Unity](#)
- [Importiere ein Beispielspiel und führe es aus](#)

Amazon GameLift ServersIntegrieren Sie in ein Unity-Gameserver-Projekt

Note

Dieses Thema enthält Informationen zu einer früheren Version des Amazon GameLift Servers Plugins für Unity. Version 1.x verwendet das Server-SDK für Amazon GameLift Servers 4.x oder früher. Dokumentation zur neuesten Plugin-Version, die das Server-SDK 5.x verwendet und neuere Funktionen wie Amazon GameLift Servers Anywhere und verwaltetes Container-Hosting unterstützt, finden Sie unter: [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#)

Dieses Thema hilft dir dabei, deinen benutzerdefinierten Spieleserver für das Hosting auf Amazon GameLift Servers vorzubereiten. Der Spieleserver muss in der Lage sein, Amazon GameLift Servers über seinen Status zu informieren, Spielsitzungen zu starten und zu beenden, wenn er dazu aufgefordert wird, und andere Aufgaben auszuführen. Weitere Informationen finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Voraussetzungen

Bevor du deinen Spieleserver integrierst, führe die folgenden Aufgaben durch:

- [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#)
- [Installieren und richten Sie das Plugin ein](#)

Richte einen neuen Serverprozess ein

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Richten Sie die Kommunikation mit dem Serverprozess ein Amazon GameLift Servers und melden Sie, dass er bereit ist, eine Spielsitzung zu veranstalten.

1. Initialisieren Sie das Server-SDK, indem Sie es aufrufen `InitSDK()`.
2. Um den Server darauf vorzubereiten, eine Spielsitzung anzunehmen, rufen Sie an `ProcessReady()` und geben Sie den Verbindungsport und den Ort der Spielsitzung an. Geben Sie die Namen der Callback-Funktionen an, die der Amazon GameLift Servers Dienst aufruft, z. B. `OnGameSession()`, `OnGameSessionUpdate()`, `OnProcessTerminate()`, `OnHealthCheck()` Amazon GameLift Servers. Es kann einige Minuten dauern, bis ein Rückruf erfolgt.
3. Amazon GameLift Servers aktualisiert den Status des Serverprozesses auf `ACTIVE`.
4. Amazon GameLift Servers ruft `onHealthCheck` regelmäßig an.

Das folgende Codebeispiel zeigt, wie Sie einen einfachen Serverprozess mit einrichten Amazon GameLift Servers.

```
//initSDK
var initSDKOutcome = GameLiftServerAPI.InitSDK();

//processReady
// Set parameters and call ProcessReady
var processParams = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    // Examples of log and error files written by the game server
    new LogParameters(new List<string>()
    {
        "C:\\game\\logs",
```

```
        "C:\\game\\error"
    ))
);

var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);

// Implement callback functions
void OnGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}

void OnProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    var ProcessEndingOutcome = GameLiftServerAPI.ProcessEnding();
}

bool OnHealthCheck()
{
    bool isHealthy;
    // complete health evaluation within 60 seconds and set health
    return isHealthy;
}
```

Starte eine Spielsitzung

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Nachdem die Initialisierung des Spiels abgeschlossen ist, können Sie eine Spielsitzung starten.

1. Implementieren Sie die Callback-Funktion. `onStartGameSession` Amazon GameLift Servers ruft diese Methode auf, um eine neue Spielsitzung auf dem Serverprozess zu starten und Spielerverbindungen zu empfangen.

2. Um eine Spielsitzung zu aktivieren, rufen Sie an. `ActivateGameSession()` Weitere Informationen zum SDK finden Sie unter [C#-Server-SDK für Amazon GameLift Servers 4.x — Aktionen](#).

Das folgende Codebeispiel zeigt, wie Sie eine Spielsitzung mit `startAmazonGameLiftServers` starten.

```
void OnStartGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    ...
    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

Beendet eine Spielsitzung

 Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Benachrichtigen Sie Amazon GameLift Servers, wenn eine Spielsitzung endet. Es hat sich bewährt, Serverprozesse nach Abschluss der Spielsitzungen herunterzufahren, um Hosting-Ressourcen zu recyceln und zu aktualisieren.

1. Richten Sie eine Funktion ein, die so benannt ist `ProcessTerminate`, dass sie Anfragen von Amazon GameLift Servers empfängt und `ProcessEnding()` aufruft.
2. Der Prozessstatus ändert sich auf `TERMINATED`.

Das folgende Beispiel beschreibt, wie ein Prozess für eine Spielsitzung beendet wird.

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();

if (processReadyOutcome.Success)
    Environment.Exit(0);

// otherwise, exit with error code
Environment.Exit(errorCode);
```

Erstellen Sie einen Server-Build und laden Sie ihn hoch auf Amazon GameLift Servers

 Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Nachdem Sie Ihren Spieleserver mit integriert haben Amazon GameLift Servers, laden Sie die Build-Dateien in eine Flotte hoch, damit sie für das Amazon GameLift Servers Hosten von Spielen eingesetzt werden können. Weitere Informationen darüber, wie du deinen Server hochlädst Amazon GameLift Servers, findest du unter [Erstelle einen Gameserver-Build für Amazon GameLift Servers](#).

Amazon GameLift Servers Integrieren Sie in ein Unity-Spieleclient-Projekt

 Note

Dieses Thema enthält Informationen zu einer früheren Version des Amazon GameLift Servers Plugins für Unity. Version 1.x verwendet das Server-SDK für Amazon GameLift Servers 4.x oder früher. Dokumentation zur neuesten Plugin-Version, die das Server-SDK 5.x verwendet und neuere Funktionen wie Amazon GameLift Servers Anywhere und verwaltetes Container-Hosting unterstützt, finden Sie unter. [Amazon GameLift Servers Plugin für Unity \(Server-SDK 5.x\)](#)

In diesem Thema erfahren Sie, wie Sie einen Spielclient einrichten, der über einen Back-End-Dienst eine Verbindung zu Amazon GameLift Servers gehosteten Spielsitzungen herstellt. Wird verwendet Amazon GameLift Servers APIs, um Spielsuche zu initiieren, die Platzierung von Spielsitzungen anzufordern und vieles mehr.

Fügen Sie dem Backend-Serviceprojekt Code hinzu, um die Kommunikation mit dem Amazon GameLift Servers Dienst zu ermöglichen. Ein Backend-Dienst kümmert sich um die gesamte Kommunikation des Spielclients mit dem GameLift Dienst. Weitere Informationen zu Backend-Diensten finden Sie unter.

Ein Backend-Server erledigt die folgenden Aufgaben des Spielclients:

- Passen Sie die Authentifizierung für Ihre Spieler an.

- Informationen über aktive Spielsitzungen vom Amazon GameLift Servers-Service anfordern.
- Erstelle eine neue Spielsitzung.
- Fügt einen Spieler zu einer bestehenden Spielsitzung hinzu.
- Entferne einen Spieler aus einer bestehenden Spielsitzung.

Themen

- [Voraussetzungen](#)
- [Initialisiere einen Spielclient](#)
- [Erstelle eine Spielsitzung für eine bestimmte Flotte](#)
- [Spieler zu Spielsitzungen hinzufügen](#)
- [Einen Spieler aus einer Spielsitzung entfernen](#)

Voraussetzungen

Bevor Sie die Spielserver-Kommunikation mit dem Amazon GameLift Servers Client einrichten, führen Sie die folgenden Aufgaben durch:

- [Richten Sie ein AWS Benutzerkonto ein](#)
- [Installieren und richten Sie das Plugin ein](#)
- [Amazon GameLift Servers integrieren Sie in ein Unity-Gameserver-Projekt](#)
- [Stellen Sie Hosting-Flotten bereit für Amazon GameLift Servers](#)

Initialisiere einen Spielclient

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Fügen Sie Code hinzu, um einen Spielclient zu initialisieren. Führe diesen Code beim Start aus, er ist für andere Amazon GameLift Servers Funktionen notwendig.

1. `InitialisierenAmazonGameLiftClient`. Rufen Sie entweder `AmazonGameLiftClient` mit einer Standard-Client-Konfiguration oder einer benutzerdefinierten Konfiguration auf. Weitere

Informationen zur Konfiguration eines Clients finden Sie unter [Richten Sie die Amazon GameLift Servers API ein](#).

2. Generieren Sie für jeden Spieler eine eindeutige Spieler-ID, um eine Verbindung zu einer Spielsitzung herzustellen. Weitere Informationen finden Sie unter [Spieler generieren IDs](#).

Die folgenden Beispiele zeigen, wie Sie einen Amazon GameLift Servers Client einrichten.

```
public class GameLiftClient
{
    private GameLift gl;
    //A sample way to generate random player IDs.
    bool includeBrackets = false;
    bool includeDashes = true;
    string playerId = AZ::Uuid::CreateRandom().ToString<string>(includeBrackets,
includeDashes);

    private Amazon.GameLift.Model.PlayerSession psession = null;
    public AmazonGameLiftClient aglc = null;

    public void CreateGameLiftClient()
    {
        //Access Amazon GameLift Servers service by setting up a configuration.
        //The default configuration specifies a location.
        var config = new AmazonGameLiftConfig();
        config.RegionEndpoint = Amazon.RegionEndpoint.USEast1;

        CredentialProfile profile = null;
        var nscf = new SharedCredentialsFile();
        nscf.TryGetProfile(profileName, out profile);
        AWSCredentials credentials = profile.GetAWSCredentials(null);
        //Initialize Amazon GameLift Servers Client with default client
configuration.
        aglc = new AmazonGameLiftClient(credentials, config);
    }
}
```

Erstelle eine Spielsitzung für eine bestimmte Flotte

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Fügen Sie Code zum Starten neuer Spielsitzungen auf den von Ihnen bereitgestellten Flotten hinzu und stellen Sie diese Spielern zur Verfügung. Nachdem Amazon GameLift Servers Sie die neue Spielsitzung erstellt und eine zurückgegeben haben `GameSession`, können Sie ihr Spieler hinzufügen.

- Stellen Sie eine Anfrage für eine neue Spielsitzung.
 - Wenn dein Spiel Flotten verwendet, rufe `CreateGameSession()` mit einer Flotten- oder Alias-ID, einem Sitzungsnamen und der maximalen Anzahl gleichzeitiger Spieler für das Spiel an.
 - Wenn dein Spiel Warteschlangen verwendet, rufe an. `StartGameSessionPlacement()`

Das folgende Beispiel zeigt, wie Sie eine Spielsitzung erstellen.

```
public Amazon.GameLift.Model.GameSession()
{
    var cgsreq = new Amazon.GameLift.Model.CreateGameSessionRequest();
    //A unique identifier for the alias with the fleet to create a game session in.
    cgsreq.AliasId = aliasId;
    //A unique identifier for a player or entity creating the game session
    cgsreq.CreatorId = playerId;
    //The maximum number of players that can be connected simultaneously to the game
    session.
    cgsreq.MaximumPlayerSessionCount = 4;

    //Prompt an available server process to start a game session and retrieves
    connection information for the new game session
    Amazon.GameLift.Model.CreateGameSessionResponse cgsres =
    aglc.CreateGameSession(cgsreq);
    string gsid = cgsres.GameSession != null ? cgsres.GameSession.GameSessionId : "N/
    A";
    Debug.Log((int)cgsres.HttpStatusCode + " GAME SESSION CREATED: " + gsid);
    return cgsres.GameSession;
```

```
}
```

Spieler zu Spielsitzungen hinzufügen

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Nachdem Amazon GameLift Servers Sie die neue Spielsitzung erstellt und ein `GameSession` Objekt zurückgegeben haben, können Sie ihr Spieler hinzufügen.

1. Reservieren Sie einen Spielerplatz in einer Spielsitzung, indem Sie eine neue Spielersitzung erstellen. Verwenden Sie `CreatePlayerSession` oder `CreatePlayerSessions` mit der Spielsitzungs-ID und einer eindeutigen ID für jeden Spieler.
2. Connect zur Spielsitzung her. Rufen Sie das `PlayerSession` Objekt ab, um die Verbindungsinformationen der Spielsitzung abzurufen. Sie können diese Informationen verwenden, um eine direkte Verbindung zum Serverprozess herzustellen:
 - a. Verwenden Sie den angegebenen Port und entweder den DNS-Namen oder die IP-Adresse des Serverprozesses.
 - b. Verwenden Sie den DNS-Namen und den Port Ihrer Flotten. Der DNS-Name und der Port sind erforderlich, wenn für Ihre Flotten die Generierung von TLS-Zertifikaten aktiviert ist.
 - c. Verweisen Sie auf die Sitzungs-ID des Players. Die Sitzungs-ID des Spielers ist erforderlich, wenn Ihr Spieleserver eingehende Spielerverbindungen validiert.

Die folgenden Beispiele zeigen, wie Sie einen Spielerplatz in einer Spielsitzung reservieren können.

```
public Amazon.GameLift.Model.PlayerSession
CreatePlayerSession(Amazon.GameLift.Model.GameSession gsession)
{
    var cpsreq = new Amazon.GameLift.Model.CreatePlayerSessionRequest();
    cpsreq.GameSessionId = gsession.GameSessionId;
    //Specify game session ID.
    cpsreq.PlayerId = playerId;
    //Specify player ID.
    Amazon.GameLift.Model.CreatePlayerSessionResponse cpsres =
    aglc.CreatePlayerSession(cpsreq);
```

```
    string psid = cpsres.PlayerSession != null ? cpsres.PlayerSession.PlayerSessionId :  
    "N/A";  
    return cpsres.PlayerSession;  
}
```

Der folgende Code zeigt, wie Sie einen Spieler mit der Spielsitzung verbinden.

```
public bool ConnectPlayer(int playerId, string playerSessionId)  
{  
    //Call ConnectPlayer with player ID and player session ID.  
    return server.ConnectPlayer(playerId, playerSessionId);  
}
```

Einen Spieler aus einer Spielsitzung entfernen

 Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Sie können die Spieler aus der Spielsitzung entfernen, wenn sie das Spiel verlassen.

1. Informiere den Amazon GameLift Servers Dienst darüber, dass ein Spieler die Verbindung zum Serverprozess getrennt hat. Rufen Sie `RemovePlayerSession` mit der Sitzungs-ID des Spielers an.
2. Stellen Sie sicher, dass das `RemovePlayerSession` zurückgegeben wird `Success`. Amazon GameLift Servers ändert dann den Spielerplatz so, dass er verfügbar ist. Amazon GameLift Servers Dieser kann einem neuen Spieler zugewiesen werden.

Das folgende Beispiel zeigt, wie eine Spielersitzung entfernt wird.

```
public void DisconnectPlayer(int playerId)  
{  
    //Receive the player session ID.  
    string playerSessionId = playerSessions[playerId];  
    var outcome = GameLiftServerAPI.RemovePlayerSession(playerSessionId);  
    if (outcome.Success)  
    {
```

```
        Debug.Log (":) PLAYER SESSION REMOVED");
    }
    else
    {
        Debug.Log(":(PLAYER SESSION REMOVE FAILED. RemovePlayerSession()
        returned " + outcome.Error.ToString());
    }
}
```

Installieren und richten Sie das Plugin ein

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

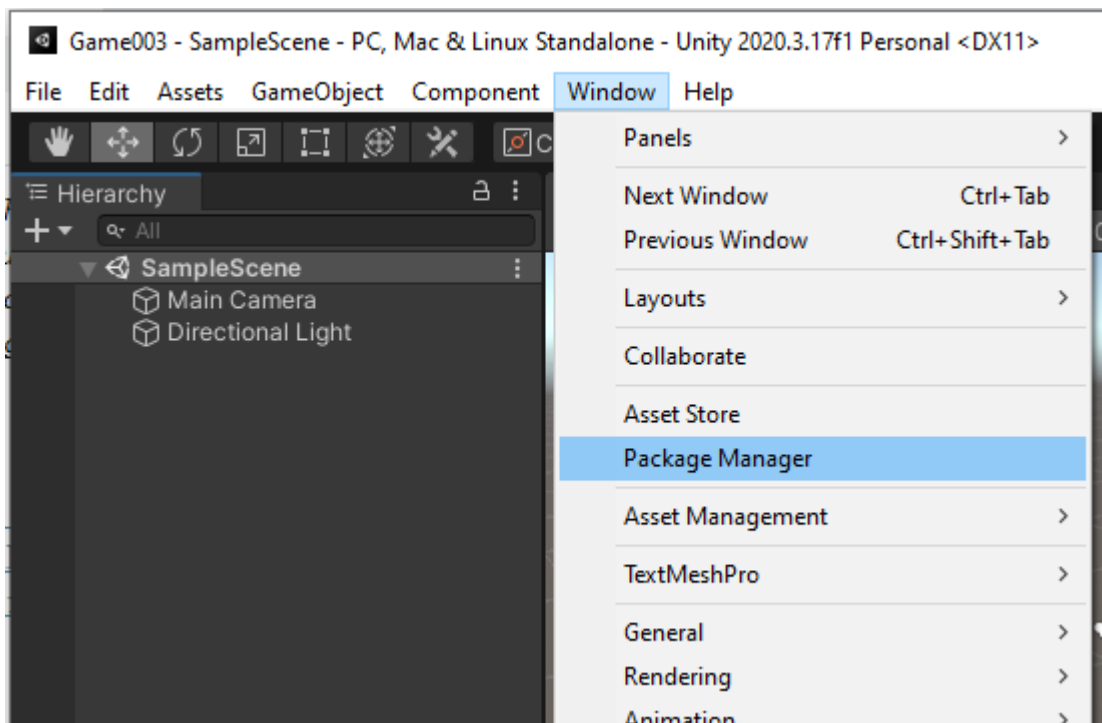
In diesem Abschnitt wird beschrieben, wie Sie das Amazon GameLift Servers Plugin für Unity, Version 1.0.0, herunterladen, installieren und einrichten.

Voraussetzungen

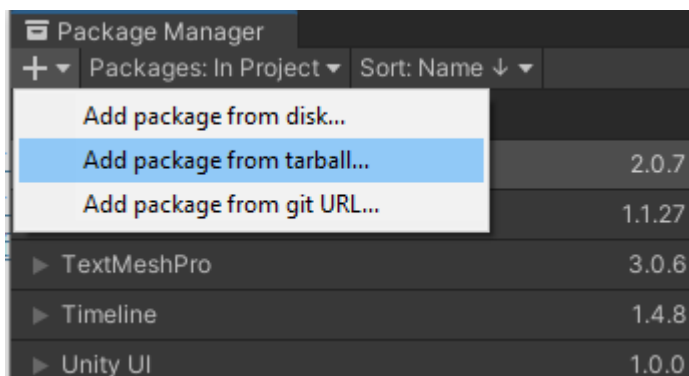
- Unity für Windows 2019.4 LTS, Windows 2020.3 LTS oder Unity für macOS
- Aktuelle Version von Java
- Aktuelle Version von .NET 4.x

Um das Plugin für Unity herunterzuladen und zu installieren

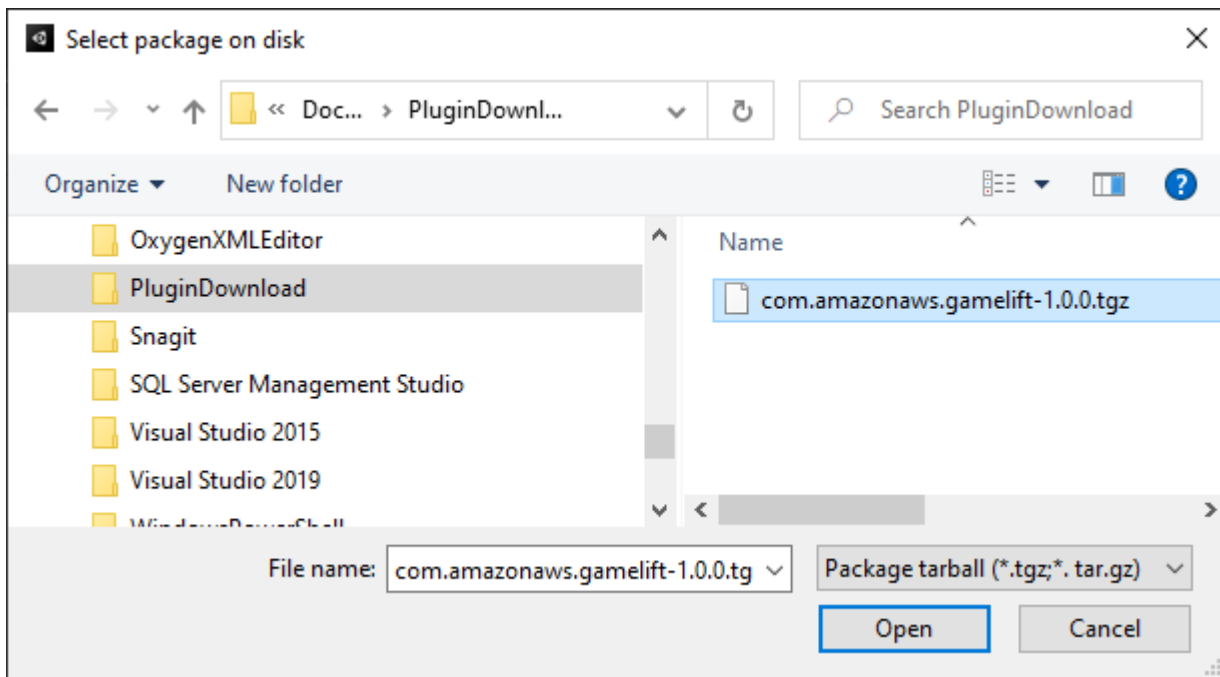
1. Laden Sie das Amazon GameLift Servers Plugin für Unity herunter. Die neueste Version finden Sie auf der [Repository-Seite des Amazon GameLift Servers Plugins für Unity](#). Wählen Sie unter der [neuesten Version](#) die Option Assets aus und laden Sie dann die `com.amazonaws.gamelift-version.tgz` Datei herunter.
2. Starten Sie Unity und wählen Sie ein Projekt aus.
3. Wählen Sie in der oberen Navigationsleiste unter Fenster die Option Package Manager aus:



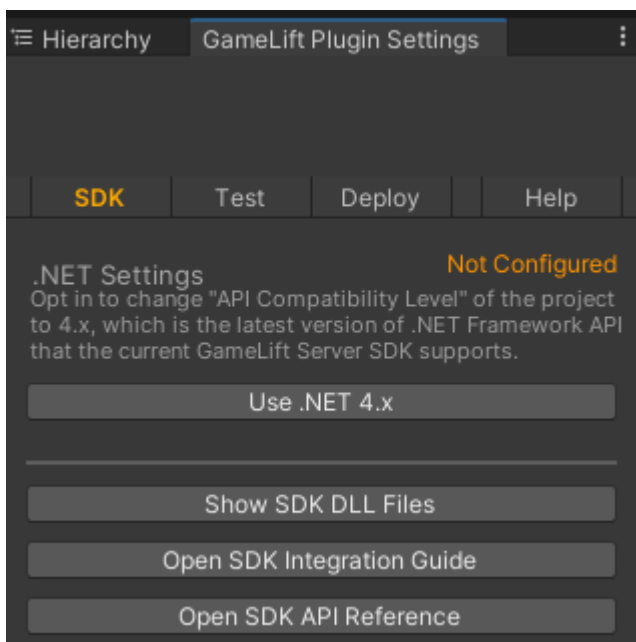
4. Wählen Sie auf der Registerkarte Paketmanager + und dann Package aus Tarball hinzufügen... :



5. Navigieren Sie im Fenster Pakete auf Festplatte auswählen zu dem `com.amazonaws.gamelift` Ordner, wählen Sie die Datei `com.amazonaws.gamelift-version.tgz` aus und klicken Sie dann auf Öffnen:



6. Nachdem Unity das Plug-in geladen hat, Amazon GameLift Servers erscheint es als neues Element im Unity-Menü. Die Installation und Neukompilierung von Skripten kann einige Minuten dauern. Die Registerkarte Amazon GameLift ServersPlugin-Einstellungen wird automatisch geöffnet.



7. Wählen Sie im SDK-Bereich die Option .NET 4.x verwenden aus.

Bei der Konfiguration ändert sich der Status von Nicht konfiguriert zu Konfiguriert.

Teste dein Spiel lokal

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Verwenden Sie Amazon GameLift Servers Local, um es Amazon GameLift Servers auf Ihrem lokalen Gerät auszuführen. Sie können Amazon GameLift Servers Local verwenden, um Codeänderungen innerhalb von Sekunden ohne Netzwerkverbindung zu überprüfen.

Konfigurieren Sie lokale Tests

1. Wählen Sie im Fenster des Plug-ins für Unity die Registerkarte Test aus.
2. Wählen Sie im Bereich Test die Option Amazon GameLift ServersLokal herunterladen aus. Das Plugin für Unity öffnet ein Browserfenster und lädt die `GameLift_06_03_2021.zip` Datei in Ihren Download-Ordner herunter.

Der Download umfasst das C#-Server-SDK, .NET-Quelldateien und eine mit Unity kompatible .NET-Komponente.

3. Entpacken Sie die heruntergeladene `GameLift_06_03_2021.zip`-Datei.
4. Wählen Sie im Fenster mit den Amazon GameLift ServersPlugin-Einstellungen die Option Amazon GameLift ServersLokaler Pfad, navigieren Sie zum entpackten Ordner, wählen Sie die Datei **GameLiftLocal.jar** aus und klicken Sie dann auf Öffnen.

Wenn konfiguriert, ändert sich der lokale Teststatus von Nicht konfiguriert zu Konfiguriert.

5. Überprüfen Sie den Status der JRE. Wenn der Status Nicht konfiguriert lautet, wählen Sie JRE herunterladen und installieren Sie die empfohlene Java-Version.

Nachdem Sie die Java-Umgebung installiert und konfiguriert haben, ändert sich der Status in Konfiguriert.

Starte dein lokales Spiel

1. Wählen Sie auf der Registerkarte „Plugin für Unity“ die Registerkarte „Test“.
2. Wählen Sie im Bereich Test die Option Open Local Test UI aus.

3. Geben Sie im Fenster Lokales Testen einen ausführbaren Serverpfad an. Wählen Sie... um den Pfad und den Namen der ausführbaren Datei Ihrer Serveranwendung auszuwählen.
4. Geben Sie im Fenster Local Testing einen lokalen GL-Port an.
5. Wählen Sie Deploy & Run, um den Server bereitzustellen und auszuführen.
6. Um deinen Gameserver zu beenden, wähle Stopp oder schließe das Spielserver-Fenster.

Stellen Sie ein Szenario bereit

 Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Ein Szenario verwendet eine CloudFormation Vorlage, um die Ressourcen zu erstellen, die Sie für die Bereitstellung einer Cloud-Hosting-Lösung für Ihr Spiel benötigen. In diesem Abschnitt werden die Amazon GameLift Servers bereitgestellten Szenarien und deren Verwendung beschrieben.

Voraussetzungen

Um das Szenario bereitzustellen, benötigen Sie eine IAM-Rolle für den Amazon GameLift Servers Dienst. Informationen zum Erstellen einer Rolle für finden Sie Amazon GameLift Servers unter [Richten Sie ein AWS Benutzerkonto ein](#).

Für jedes Szenario sind Berechtigungen für die folgenden Ressourcen erforderlich:

- Amazon GameLift Servers
- Amazon S3
- CloudFormation
- API Gateway
- AWS Lambda
- AWS WAFV2
- Amazon Cognito

Szenarien

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Das Amazon GameLift Servers Plug-in für Unity umfasst die folgenden Szenarien:

Nur Auth

In diesem Szenario wird ein Spiele-Backend-Dienst erstellt, der die Spielerauthentifizierung ohne Spielserverfunktion durchführt. Die Vorlage erstellt die folgenden Ressourcen in Ihrem Konto:

- Ein Amazon Cognito Cognito-Benutzerpool zum Speichern von Spielerauthentifizierungsinformationen.
- Ein endpunktgestützter AWS Lambda REST-Handler von Amazon API Gateway, der Spiele startet und Verbindungsinformationen zu Spielen anzeigt.

Flotte mit nur einer Region

In diesem Szenario wird ein Spiel-Backend-Service mit einer einzigen Amazon GameLift Servers Flotte erstellt. Es erstellt die folgenden Ressourcen:

- Ein Amazon Cognito Cognito-Benutzerpool, über den sich ein Spieler authentifizieren und ein Spiel starten kann.
- Ein AWS Lambda Handler zur Suche nach einer bestehenden Spielsitzung mit einem offenen Spielerplatz auf der Flotte. Wenn es keinen offenen Slot finden kann, erstellt es eine neue Spielsitzung.

Flotte mit mehreren Regionen, Warteschlange und benutzerdefiniertem Matchmaker

In diesem Szenario werden Spiele mithilfe von Amazon GameLift Servers Warteschlangen und einem benutzerdefinierten Matchmaker gebildet, um die ältesten Spieler im Wartepool zu gruppieren. Es erstellt die folgenden Ressourcen:

- Ein Thema von Amazon Simple Notification Service, das Nachrichten Amazon GameLift Servers veröffentlicht an. Weitere Informationen zu SNS-Themen und Benachrichtigungen finden Sie unter [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#).
- Eine Lambda-Funktion, die durch die Nachricht aufgerufen wird, die die Platzierungs- und Spielverbindungsdetails übermittelt.
- Eine Amazon DynamoDB-Tabelle zum Speichern von Platzierungs- und Spielverbindungsdetails. `GetGameConnection` Aufrufe lesen aus dieser Tabelle und geben die Verbindungsinformationen an den Spielclient zurück.

Finde Flotten mit einer Warteschlange und einem benutzerdefinierten Matchmaker

In diesem Szenario werden mithilfe von Amazon GameLift Servers Warteschlangen und einem benutzerdefinierten Matchmaker Matches gebildet und drei Flotten konfiguriert. Es erstellt die folgenden Ressourcen:

- Zwei Spot-Flotten, die unterschiedliche Instance-Typen enthalten, um Stabilität bei Nichtverfügbarkeit von Spot zu gewährleisten.
- Eine On-Demand-Flotte, die als Backup für die anderen Spot-Flotten dient. Weitere Informationen zur Gestaltung Ihrer Flotten finden Sie unter [Anpassungen der Hosting-Ressourcen](#)
- Eine Amazon GameLift Servers Warteschlange, um die Serververfügbarkeit hoch und die Kosten niedrig zu halten. Weitere Informationen und bewährte Methoden zu Warteschlangen finden Sie unter [Passen Sie eine Warteschlange für Spielsitzungen an](#).

FlexMatch

In diesem Szenario wird ein verwalteter Matchmaking-Dienst verwendet FlexMatch, um Spieler zusammenzubringen. Weitere Informationen zu finden Sie FlexMatch unter [Was ist Amazon GameLift Servers FlexMatch](#). In diesem Szenario werden die folgenden Ressourcen erstellt:

- Eine Lambda-Funktion zum Erstellen eines Matchmaking-Tickets, nachdem Anfragen eingegangen sind. `StartGame`
- Eine separate Lambda-Funktion zum Abhören von FlexMatch Match-Ereignissen.

Um unnötige Kosten für Sie zu vermeiden AWS-Konto, entfernen Sie die Ressourcen, die durch jedes Szenario erstellt wurden, nachdem Sie sie nicht mehr verwendet haben. Löschen Sie den entsprechenden CloudFormation Stapel.

AWS Anmeldeinformationen aktualisieren

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Das Amazon GameLift Servers Plugin für Unity benötigt Sicherheitsanmeldedaten, um ein Szenario bereitzustellen. Sie können entweder neue Anmeldeinformationen erstellen oder vorhandene Anmeldeinformationen verwenden.

Weitere Informationen zur Konfiguration von Anmeldeinformationen finden Sie unter [Grundlegendes zu Ihren AWS Anmeldeinformationen und zum Abrufen von Anmeldeinformationen](#).

So aktualisieren Sie Ihre AWS Anmeldeinformationen

1. Wählen Sie in Unity auf der Registerkarte Plugin für Unity die Registerkarte Bereitstellen aus.
2. Wählen Sie im Bereich Bereitstellen die Option AWS Credentials aus.
3. Sie können neue AWS Anmeldeinformationen erstellen oder vorhandene Anmeldeinformationen auswählen.
 - Um Anmeldeinformationen zu erstellen, wählen Sie Neues Anmeldeinformationsprofil erstellen aus und geben Sie dann den neuen Profilnamen, die AWS Zugriffsschlüssel-ID, den AWS geheimen Schlüssel und an AWS-Region.
 - Um vorhandene Anmeldeinformationen auszuwählen, wählen Sie Bestehendes Anmeldeinformationsprofil auswählen und wählen Sie dann einen Profilnamen und AWS-Region.
4. Wählen Sie im Fenster „AWS Anmeldeinformationen aktualisieren“ die Option „Anmeldedatenprofil aktualisieren“.

Aktualisieren Sie den Konto-Bootstrap

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Der Bootstrap-Speicherort ist ein Amazon S3 S3-Bucket, der während der Bereitstellung verwendet wird. Es wird verwendet, um Spieleserver-Assets und andere Abhängigkeiten zu speichern. Die Region, die AWS-Region Sie für den Bucket auswählen, muss dieselbe Region sein, die Sie für die Szenariobereitstellung verwenden werden.

Weitere Informationen zu Amazon S3 S3-Buckets finden Sie unter [Amazon Simple Storage Service-Buckets erstellen, konfigurieren und damit arbeiten](#).

Um den Bootstrap-Speicherort des Kontos zu aktualisieren

1. Wählen Sie in Unity auf der Registerkarte Plugin für Unity die Registerkarte Bereitstellen aus.
2. Wählen Sie im Bereich „Bereitstellen“ die Option „Account Bootstrap aktualisieren“.
3. Im Fenster Account Bootstrapping wählen Sie einen vorhandenen Amazon S3 S3-Bucket aus oder erstellen einen neuen Amazon S3 S3-Bucket:
 - Um einen vorhandenen Bucket auszuwählen, wählen Sie Bestehenden Amazon S3 S3-Bucket auswählen und Aktualisieren, um Ihre Auswahl zu speichern.
 - Wählen Sie Neuen Amazon S3 S3-Bucket erstellen, um einen neuen Amazon Simple Storage Service-Bucket zu erstellen, und wählen Sie dann eine Richtlinie aus. Die Richtlinie legt fest, wann der Amazon S3 S3-Bucket abläuft. Wählen Sie Create, um den Bucket zu erstellen.

Stellen Sie ein Spielszenario bereit

 Note

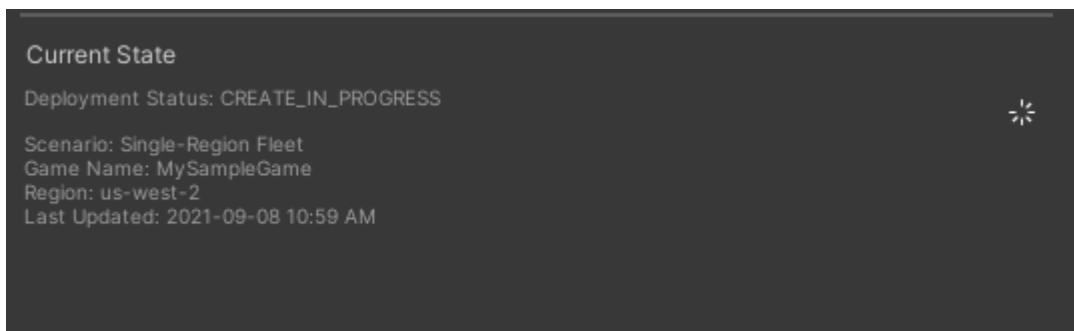
Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Sie können ein Szenario verwenden, um Ihr Spiel zu testen. Amazon GameLift Servers Jedes Szenario verwendet eine CloudFormation Vorlage, um einen Stack mit den erforderlichen Ressourcen zu erstellen. Die meisten Szenarien erfordern eine ausführbare Datei und einen Build-Pfad für den Spieleserver. Wenn Sie das Szenario bereitstellen, werden die Spielressourcen im Rahmen der Bereitstellung an den Bootstrap-Speicherort Amazon GameLift Servers kopiert.

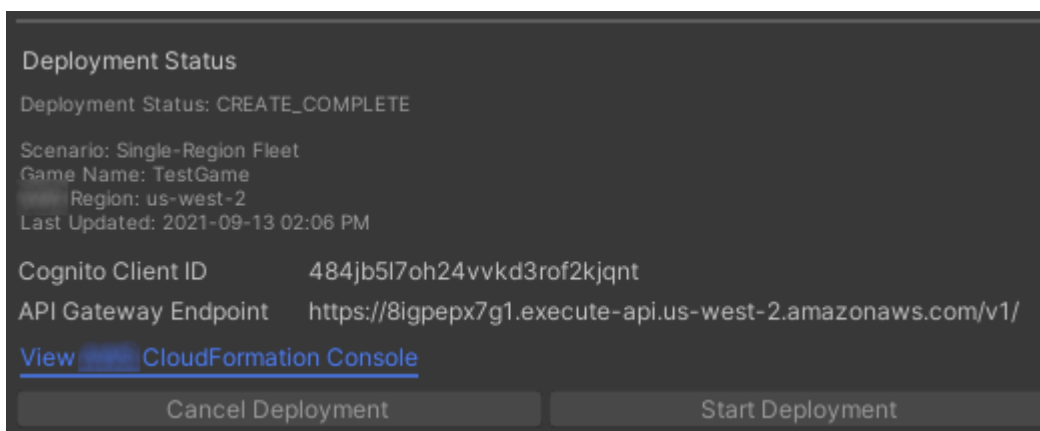
Sie müssen AWS Anmeldeinformationen und einen AWS Konto-Bootstrap konfigurieren, um ein Szenario bereitzustellen.

Um ein Szenario bereitzustellen

1. Wählen Sie in Unity auf der Registerkarte Plugin für Unity die Registerkarte Bereitstellen aus.
2. Wählen Sie im Bereich „Bereitstellen“ die Option „Bereitstellungsoberfläche öffnen“.
3. Wählen Sie im Bereitstellungsfenster ein Szenario aus.
4. Geben Sie einen Spielnamen ein. Dieser Wert muss eindeutig sein. Der Spielname ist Teil des CloudFormation Stack-Namens, wenn Sie das Szenario bereitstellen.
5. Wählen Sie den Pfad zum Build-Ordner für den Gameserver. Der Pfad zum Build-Ordner verweist auf den Ordner, der die ausführbare Serverdatei und die Abhängigkeiten enthält.
6. Wählen Sie den Game Server Build .exe-Dateipfad. Der Pfad der ausführbaren Build-Datei verweist auf die ausführbare Datei des Spielservers.
7. Wählen Sie „Bereitstellung starten“, um mit der Bereitstellung eines Szenarios zu beginnen. Sie können den Status des Updates im Fenster Bereitstellung unter Aktueller Status verfolgen. Die Bereitstellung von Szenarien kann mehrere Minuten dauern.



8. Wenn die Bereitstellung des Szenarios abgeschlossen ist, wird der aktuelle Status aktualisiert und enthält nun die Cognito-Client-ID und den API-Gateway-Endpunkt, die Sie kopieren und in das Spiel einfügen können.



9. Um die Spieleinstellungen zu aktualisieren, wählen Sie im Unity-Menü die Option Gehe zu Client-Verbindungseinstellungen. Dadurch wird auf der rechten Seite des Unity-Bildschirms ein Inspektor-Tab angezeigt.
10. Deaktivieren Sie den lokalen Testmodus.
11. Geben Sie den API Gateway Endpoint und die Cognito Client ID ein. Wählen Sie dasselbe aus, das AWS-Region Sie für die Szenariobereitstellung verwendet haben. Anschließend kannst du den Spielclient mithilfe der bereitgestellten Szenario-Ressourcen neu erstellen und ausführen.

Löschen von Ressourcen, die durch das Szenario erstellt wurden

 Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Um die für das Szenario erstellten Ressourcen zu löschen, löschen Sie den entsprechenden CloudFormation Stack.

Um Ressourcen zu löschen, die durch das Szenario erstellt wurden

1. Wählen Sie im Fenster des Amazon GameLift Servers Plug-ins für Unity Deployment die Option View AWS CloudFormation Console aus, um die CloudFormation Konsole zu öffnen.
2. Wählen Sie in der CloudFormation Konsole Stacks und dann den Stack aus, der den bei der Bereitstellung angegebenen Spielnamen enthält.
3. Wähle Löschen, um den Stapel zu löschen. Das Löschen eines Stacks kann einige Minuten dauern. Nachdem der vom Szenario verwendete Stack CloudFormation gelöscht wurde, ändert sich sein Status auf ROLLBACK_COMPLETE.

Integrieren Sie Spiele Amazon GameLift Servers in Unity

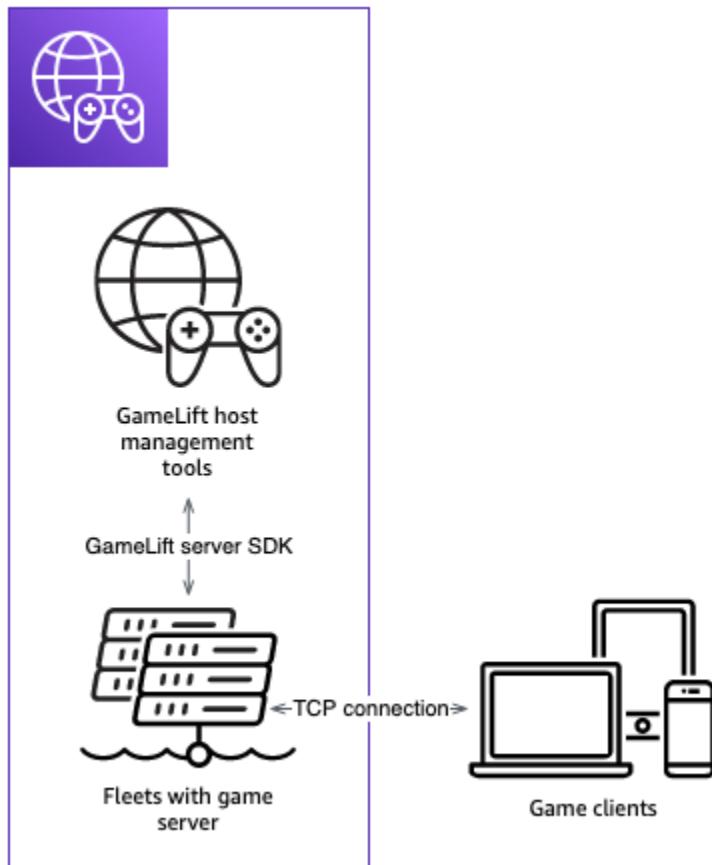
 Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Integrieren Sie Ihr Unity-Spiel in, Amazon GameLift Servers indem Sie die folgenden Aufgaben erledigen:

- [Amazon GameLift ServersIntegrieren Sie in ein Unity-Gameserver-Projekt](#)
- [Amazon GameLift ServersIntegrieren Sie in ein Unity-Spieleclient-Projekt](#)

Das folgende Diagramm zeigt ein Beispiel für den Ablauf der Integration eines Spiels. In der Abbildung wird eine Flotte mit dem Spielserver bereitgestelltAmazon GameLift Servers. Der Spielclient kommuniziert mit dem Spielserver, der mit kommuniziert. Amazon GameLift Servers



Importiere ein Beispielspiel und führe es aus

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Das Amazon GameLift Servers Plugin für Unity enthält ein Beispielspiel, mit dem Sie die Grundlagen der Integration Ihres Spiels erkunden können. Amazon GameLift Servers In diesem Abschnitt erstellt du den Spielclient und den Spieleserver und testest dann lokal mit Amazon GameLift Servers Local.

Voraussetzungen

- [Richten Sie ein AWS Benutzerkonto ein](#)
- [Installieren und richten Sie das Plugin ein](#)

Erstelle den Beispiel-Spieler und führe ihn aus

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Richten Sie die Spielserverdateien des Beispielspiels ein.

1. Wählen Sie Amazon GameLift Servers in Unity im Menü die Option Beispielspiel importieren aus.
2. Wählen Sie im Fenster „Beispielspiel importieren“ die Option „Importieren“, um das Spiel, seine Inhalte und Abhängigkeiten zu importieren.
3. Baue den Spieleserver. Wählen Sie Amazon GameLift Servers in Unity im Menü die Option und dann Apply Windows Sample Server Build Settings oder Apply macOS Sample Server Build Settings aus. Nachdem Sie die Spielserver-Einstellungen konfiguriert haben, kompiliert Unity die Assets neu.
4. Wählen Sie in Unity im Menü Datei und dann Build aus. Wählen Sie Server Build, dann Build und dann einen Build-Ordner speziell für Serverdateien.

Unity erstellt den Beispielspieler und platziert die ausführbare Datei und die erforderlichen Ressourcen im angegebenen Build-Ordner.

Erstellen Sie den Beispiel-Spielclient und führen Sie ihn aus

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Richte die Spielclient-Dateien des Beispielspiels ein.

1. Wählen Sie Amazon GameLift Servers in Unity im Menü die Option und dann Apply Windows Sample Client Build Settings oder Apply macOS Sample Client Build Settings aus. Nachdem die Einstellungen des Spielclients konfiguriert wurden, kompiliert Unity die Assets neu.
2. Wählen Sie in Unity im Menü die Option Gehe zu Client-Einstellungen aus. Dadurch wird auf der rechten Seite des Unity-Bildschirms ein Inspektor-Tab angezeigt. Wählen Sie auf der Registerkarte Amazon GameLift Servers Client-Einstellungen die Option Lokaler Testmodus aus.
3. Baue den Spielclient. Wählen Sie in Unity im Menü die Option Datei. Vergewissern Sie sich, dass Server Build nicht aktiviert ist, wählen Sie Build und wählen Sie dann einen Build-Ordner speziell für Client-Dateien aus.

Unity erstellt den Beispiel-Spielclient und platziert die ausführbare Datei und die erforderlichen Ressourcen im angegebenen Client-Build-Ordner.

4. Sie haben den Spielserver und den Client noch nicht erstellt. In den nächsten Schritten führst du das Spiel aus und siehst, wie es interagiert mit Amazon GameLift Servers.

Testen Sie das Beispielspiel lokal

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Führen Sie das Beispielspiel aus, das Sie mit Amazon GameLift Servers Local importiert haben.

1. Starte den Spieleserver. Wählen Sie in Unity auf der Registerkarte Plugin für Unity die Registerkarte Bereitstellen.
2. Wählen Sie im Bereich Test die Option Open Local Test UI aus.

3. Geben Sie im Fenster Lokales Testen einen Pfad für die EXE-Datei für den Game Server an. Der Pfad muss den Namen der ausführbaren Datei enthalten. Beispiel, `C:/MyGame/GameServer/MyGameServer.exe`.
4. Wählen Sie Deploy and Run. Das Plugin für Unity startet den Spieleserver und öffnet ein Amazon GameLift Servers lokales Protokollfenster. Das Fenster enthält Protokollnachrichten, einschließlich Nachrichten, die zwischen dem Spieleserver und Amazon GameLift Servers Local gesendet wurden.
5. Starte den Spielclient. Suchen Sie mit dem Beispiel-Spielclient den Build-Speicherort und wählen Sie die ausführbare Datei aus.
6. Gib im Amazon GameLift ServersBeispielspiel eine E-Mail-Adresse und ein Passwort ein und wähle dann „Anmelden“. Die E-Mail-Adresse und das Passwort wurden nicht validiert oder verwendet.
7. Wähle im Amazon GameLift ServersBeispielspiel die Option Start aus. Der Spielclient sucht nach einer Spielsitzung. Wenn er keine Sitzung finden kann, erstellt er eine. Der Spielclient startet dann die Spielsitzung. Sie können die Spielaktivitäten in den Protokollen sehen.

Beispiele für Spielserverprotokolle

```
...
2021-09-15T19:55:3495 PID:20728 Log :) GAMELIFT AWAKE
2021-09-15T19:55:3512 PID:20728 Log :) I AM SERVER
2021-09-15T19:55:3514 PID:20728 Log :) GAMELIFT StartServer at port 33430.
2021-09-15T19:55:3514 PID:20728 Log :) SDK VERSION: 4.0.2
2021-09-15T19:55:3556 PID:20728 Log :) SERVER IS IN A GAMELIFT FLEET
2021-09-15T19:55:3577 PID:20728 Log :) PROCESSREADY SUCCESS.
2021-09-15T19:55:3577 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
...
2021-09-15T19:55:3634 PID:20728 Log :) GAMELOGIC AWAKE
2021-09-15T19:55:3635 PID:20728 Log :) GAMELOGIC START
2021-09-15T19:55:3636 PID:20728 Log :) LISTENING ON PORT 33430
2021-09-15T19:55:3636 PID:20728 Log SERVER: Frame: 0 HELLO WORLD!
...
2021-09-15T19:56:2464 PID:20728 Log :) GAMELIFT SESSION REQUESTED
2021-09-15T19:56:2468 PID:20728 Log :) GAME SESSION ACTIVATED
2021-09-15T19:56:3578 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
2021-09-15T19:57:3584 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
2021-09-15T19:58:0334 PID:20728 Log SERVER: Frame: 8695 Connection accepted: playerId
0 joined
```

```

2021-09-15T19:58:0335 PID:20728 Log SERVER: Frame: 8696 Connection accepted: playerId
1 joined
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerId 0 Msg:
CONNECT: server IP localhost
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 0:CONNECT:
server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 0
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerId 1 Msg:
CONNECT: server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 1:CONNECT:
server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 1

```

Beispiele für Amazon GameLift Servers lokale Protokolle

```

12:55:26,000 INFO || - [SocketIOServer] main - Session store / pubsub factory used:
MemoryStoreFactory (local session store only)
12:55:28,092 WARN || - [ServerBootstrap] main - Unknown channel option 'SO_LINGER' for
channel '[id: 0xe23d0a14]'
12:55:28,101 INFO || - [SocketIOServer] nioEventLoopGroup-2-1 - SocketIO server
started at port: 5757
12:55:28,101 INFO || - [SDKConnection] main - GameLift SDK server (communicates with
your game server) has started on http://localhost:5757
12:55:28,120 INFO || - [SdkWebSocketServer] WebSocketSelector-20 - WebSocket Server
started on address localhost/127.0.0.1:5759
12:55:28,166 INFO || - [StandAloneServer] main - GameLift Client server (listens for
GameLift client APIs) has started on http://localhost:8080
12:55:28,179 INFO || - [StandAloneServer] main - GameLift server sdk http listener has
started on http://localhost:5758
12:55:35,453 INFO || - [SdkWebSocketServer] WebSocketWorker-12 - onOpen
socket: /?pID=20728&sdkVersion=4.0.2&sdkLanguage=CSharp and handshake /?
pID=20728&sdkVersion=4.0.2&sdkLanguage=CSharp
12:55:35,551 INFO || - [HostProcessManager] WebSocketWorker-12 - client connected with
pID 20728
12:55:35,718 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
GameLift API to use: ProcessReady for pId 20728
12:55:35,718 INFO || - [ProcessReadyHandler] GameLiftSdkHttpHandler-thread-0 -
Received API call for processReady from 20728
12:55:35,738 INFO || - [ProcessReadyHandler] GameLiftSdkHttpHandler-thread-0 -
onProcessReady data: port: 33430
12:55:35,739 INFO || - [HostProcessManager] GameLiftSdkHttpHandler-thread-0 -
Registered new process with pId 20728

```

```

12:55:35,789 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
  GameLift API to use: ReportHealth for pId 20728
12:55:35,790 INFO || - [ReportHealthHandler] GameLiftSdkHttpHandler-thread-0 -
  Received API call for ReportHealth from 20728
12:55:35,794 INFO || - [ReportHealthHandler] GameLiftSdkHttpHandler-thread-0 -
  ReportHealth data: healthStatus: true
12:56:24,098 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
  GameLift.DescribeGameSessions
12:56:24,119 INFO || - [DescribeGameSessionsDispatcher] Thread-12 - Received API call
  to describe game sessions with input: {"FleetId":"fleet-123"}
12:56:24,241 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
  GameLift.CreateGameSession
12:56:24,242 INFO || - [CreateGameSessionDispatcher] Thread-12 - Received API call to
  create game session with input: {"FleetId":"fleet-123","MaximumPlayerSessionCount":4}
12:56:24,265 INFO || - [HostProcessManager] Thread-12 - Reserved process:
  20728 for gameSession: arn:aws:gamelift:local::gamesession/fleet-123/
  gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d
12:56:24,266 INFO || - [WebSocketInvoker] Thread-12 - StartGameSessionRequest:
  gameSessionId=arn:aws:gamelift:local::gamesession/fleet-123/
  gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d, fleetId=fleet-123, gameSessionName=null,
  maxPlayers=4, properties=[], ipAddress=127.0.0.1, port=33430, gameSessionData?=false,
  matchmakerData?=false, dnsName=localhost
12:56:24,564 INFO || - [CreateGameSessionDispatcher] Thread-12 - GameSession with
  id: arn:aws:gamelift:local::gamesession/fleet-123/gsess-59f6cc44-4361-42f5-95b5-
  fdb5825c0f3d created
12:56:24,585 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
  GameLift.DescribeGameSessions
12:56:24,585 INFO || - [DescribeGameSessionsDispatcher] Thread-12 - Received API call
  to describe game sessions with input: {"FleetId":"fleet-123"}
12:56:24,660 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
  GameLift API to use: GameSessionActivate for pId 20728
12:56:24,661 INFO || - [GameSessionActivateHandler] GameLiftSdkHttpHandler-thread-0 -
  Received API call for GameSessionActivate from 20728
12:56:24,678 INFO || - [GameSessionActivateHandler] GameLiftSdkHttpHandler-thread-0
  - GameSessionActivate data: gameSessionId: "arn:aws:gamelift:local::gamesession/
  fleet-123/gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d"

```

Serverprozess herunterfahren

Note

Dieses Thema bezieht sich auf das Amazon GameLift Servers Plugin für Unity Version 1.0.0, das das Server-SDK 4.x oder früher verwendet.

Wenn Sie mit Ihrem Beispielspiel fertig sind, fahren Sie den Server in Unity herunter.

1. Wählen Sie im Spielclient „Beenden“ oder schließen Sie das Fenster, um den Spielclient zu beenden.
2. Wählen Sie in Unity im Fenster Lokales Testen die Option Stopp oder schließen Sie das Spielserverfenster, um den Server zu beenden.

Integrieren Sie einen Spieleserver mit Amazon GameLift Servers

Ihr benutzerdefinierter Spieleserver muss in der Lage sein, mit diesen zu interagierenAmazon GameLift Servers, wenn er bereitgestellt wird und auf Hosting-Ressourcen läuft. In den Themen dieses Abschnitts wird beschrieben, wie Sie Ihr Gameserver-Projekt in das Server-SDK integrierenAmazon GameLift Servers, um erforderliche und optionale Funktionen hinzuzufügen. Dieser Abschnitt enthält allgemeine Anweisungen, die Sie mit jedem sprachspezifischen Server verwenden können. SDKs Er enthält auch detaillierte Anleitungen zur Integration Ihrer Unreal Engine- und Unity-Spieleprojekte.

Themen

- [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#)
- [Integrieren Sie das Server-SDK für Game-Engines](#)

Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu

Dieses Thema enthält allgemeine Hinweise zu den Server-SDK-Funktionen, die Sie Ihrem Spieleserver-Code hinzufügen müssen/Diese Funktion ermöglicht es Spielserverprozessen, mit dem Amazon GameLift Servers Dienst zu kommunizieren. Verwenden Sie diese Anweisungen für Spieleserver, die Sie auf Amazon GameLift Servers verwalteten EC2-Flotten, verwalteten Container-Flotten oder Anywhere-Flotten bereitstellen möchten.

Spielerprozess kommunizieren mit dem Amazon GameLift Servers Dienst, um Anweisungen vom Dienst zu erhalten und den Zustand der Serverprozesse und den Status der Spielsitzung zu melden. Ausführliche Informationen zu Interaktionen zwischen den Komponenten deiner Game-Hosting-Lösung (Spieler, Backend-Dienst, Spieleclient und Amazon GameLift Servers) findest du unter [client/server Spielinteraktionen mit Amazon GameLift Servers](#).

Holen Sie sich das Server-SDK

Um dein Spiel für das Hosting vorzubereiten, füge das Server-SDK für Amazon GameLift Servers zu deinem Gameserver-Projekt hinzu. Das Server-SDK ist in mehreren Sprachen verfügbar. Weitere Informationen zur Toolunterstützung für Spieler, einschließlich des Server-SDK, finden Sie unter [Holen Sie sich Amazon GameLift Servers Entwicklungstools](#).

Laden Sie das Server-SDK für Ihre Entwicklungssprache von github.com/amazon-gamelift herunter. Sie können Repos für die Server-SDK-Versionen, Plugins und andere Entwicklungstools finden.

Note

Wenn du mit Unreal Engine oder Unity entwickelst, lade das Amazon GameLift Servers Plugin für diese Game-Engines herunter. Das Server-SDK ist integriert und sofort einsatzbereit. Weitere Informationen finden Sie unter diesen Links:

- Unreal Engine ([Plugin herunterladen](#)) ([Integrationsanleitung](#))
- Unity ([Plugin herunterladen](#)) ([Integrationsanleitung](#))

API-Referenzen für das Server-SDK:

- [C++-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- [Go-Server-SDK für Amazon GameLift Servers -- Aktionen](#)

Initialisieren Sie den Serverprozess

Fügen Sie Code hinzu, um die Kommunikation mit dem Amazon GameLift Servers Dienst herzustellen und zu melden, wenn der Spieler-Server-Prozess bereit ist, eine Spielsitzung zu hosten. Dieser Code muss vor jedem Amazon GameLift Servers Code ausgeführt werden.

1. Initialisieren Sie einen Amazon GameLift Servers API-Client durch Aufrufen `InitSdk()`. [Wenn du deinen Spieleserver für die Ausführung auf Amazon GameLift Servers verwalteten EC2-Flotten vorbereitest, verwende den Standard `InitSDK\(\)` \(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\) ohne Parameter](#). Der API-Client kümmert sich für Sie um die Verbindung zum Dienst. Amazon GameLift Servers

 Wenn du deinen Spieleserver für den Einsatz auf einer Amazon GameLift Servers Anywhere-Flotte vorbereitest:

Initialisieren Sie den Amazon GameLift Servers API-Client, indem Sie ihn wie folgt `ServerParameters` aufrufen `InitSdk()`:

- Die URL des Websockets, der für die Verbindung zu deinem Spieleserver verwendet wurde.
- Die ID des Prozesses, der zum Hosten deines Spieleservers verwendet wird.
- Die ID des Computers, auf dem deine Gameserver-Prozesse gehostet werden.
- Die ID der Flotte, die Ihren Amazon GameLift Servers Anywhere-Computer enthält.
- Das durch den Amazon GameLift Servers Vorgang generierte Autorisierungstoken [GetComputeAuthToken](#).

2. Informiere den Dienst, dass der Spieleserver-Prozess bereit ist, eine Spielsitzung zu hosten. Rufen Sie `ProcessReady()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) ([C++](#)) wie folgt auf. `ProcessParameters` Jeder Spieleserver-Prozess darf nur einmal aufgerufen werden. `ProcessReady()`

- Die Portnummer für den Serverprozess. Wenn der Serverprozess eine Spielsitzung startet, stellt er den Port für den Amazon GameLift Servers Dienst bereit, der die Informationen zur Spielsitzung aktualisiert. Ihr Spiel kann diese Informationen abrufen und an Spieleclients weitergeben, die sie verwenden, um sich mit dem Serverprozess zu verbinden und an der Spielsitzung teilzunehmen.
- Die Speicherorte der Dateien, die Sie für Sie speichern möchten Amazon GameLift Servers. Dazu können Spielsitzungsprotokolle und andere Dateien gehören, die der Serverprozess während einer Spielsitzung generiert. Diese Dateien werden zwar vorübergehend auf dem Computer Amazon GameLift Servers gespeichert, auf dem der Serverprozess ausgeführt wird, sie sind jedoch nur verfügbar, bis die Instanz heruntergefahren wird. Sie können über die [Amazon GameLift Servers Konsole](#) oder durch Aufrufen der Amazon GameLift Servers API-Operation [GetGameSessionLogUrl\(\)](#) auf gespeicherte Dateien zugreifen.

- i** Wenn du deinen Spielseserver für den Einsatz in einer Amazon GameLift Servers verwalteten Containerflotte vorbereitest:

Du musst keine Log-Parameter für eine Containerflotte angeben. Senden Sie stattdessen die Spielsitzung und andere Protokolldaten an die Standardausgabe. Containerflotten erfassen automatisch die gesamte Container-Standardausgabe als Protokollstream.

- Die folgenden Callback-Funktionen, die es ermöglichen Amazon GameLift Servers, Nachrichten oder Eingabeaufforderungen an einen Spielseserver-Prozess zu senden. Sie müssen jede dieser Funktionen in Ihrem Spielseservercode implementieren. [Weitere Informationen finden Sie unter `ProcessParameters\(C++\)` \(C#\) \(Unreal\) \(Go\) \(C++\)](#)
- (Optional) `onHealthCheck` — Amazon GameLift Servers ruft diese Funktion regelmäßig auf, um einen Statusbericht vom Server anzufordern.
- `onStartGameSession` — Amazon GameLift Servers ruft diese Funktion als Antwort auf die Client-Anfrage [CreateGameSession\(\)](#) auf.
- `onProcessTerminate` — Amazon GameLift Servers zwingt den Serverprozess zum Beenden, sodass er ordnungsgemäß heruntergefahren wird.
- (Optional) `onUpdateGameSession` — Amazon GameLift Servers übermittelt ein aktualisiertes Objekt für die Spielsitzung an den Spielseserver oder informiert über eine Statusaktualisierung bei einer Anfrage zum Auffüllen eines Matches. Für die [FlexMatchBackfill-Funktion](#) ist dieser Callback erforderlich.

Du kannst auch einen Spielseserver einrichten, sodass dieser sicher auf andere AWS Ressourcen zugreifen kann, die dir gehören oder die du kontrollierst. Weitere Informationen finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spielseserver mit anderen AWS Ressourcen](#).

(Optional) Melden Sie den Zustand des Serverprozesses

Füge deinem Spielseserver Code hinzu, um die Callback-Funktion `onHealthCheck()` zu implementieren. Amazon GameLift Servers ruft diese Callback-Methode regelmäßig auf, um Gesundheitsmetriken zu sammeln. Gehen Sie wie folgt vor, um diese Rückruffunktion zu implementieren:

- Bewerten Sie den Integritätsstatus des Serverprozesses. Beispielsweise können Sie den Serverprozess als fehlerhaft melden, wenn externe Abhängigkeiten ausgefallen sind.
- Schließen Sie die Zustandsprüfung ab, und antworten Sie auf den Callback innerhalb von 60 Sekunden. Wenn Amazon GameLift Servers innerhalb dieser Zeit keine Antwort eingeht, wird der Serverprozess automatisch als fehlerhaft eingestuft.
- Gibt einen booleschen Wert zurück: true für fehlerfrei, false für ungesund.

Wenn Sie keinen Rückruf zur Integritätsprüfung implementieren, wird der Serverprozess als fehlerfrei Amazon GameLift Servers betrachtet, sofern der Server nicht reagiert.

Der Amazon GameLift Servers Dienst verwendet die Integrität von Serverprozessen, um fehlerhafte Prozesse zu beenden und Ressourcen zu löschen. Wenn ein Serverprozess bei drei aufeinanderfolgenden Integritätsprüfungen weiterhin als fehlerhaft gemeldet wird oder nicht reagiert, kann der Dienst den Prozess beenden und einen neuen starten. Der Service sammelt Messwerte zum Zustand der Serverprozesse einer Flotte.

(Optional) Besorgen Sie sich ein TLS-Zertifikat

Wenn der Serverprozess auf einer Flotte läuft, für die die TLS-Zertifikatsgenerierung aktiviert ist, können Sie das TLS-Zertifikat abrufen, um eine sichere Verbindung mit einem Spielclient herzustellen und die Client-Server-Kommunikation zu verschlüsseln. Eine Kopie des Zertifikats wird auf der Instance gespeichert. [Rufen Sie \(C++\) \(C#\) GetComputeCertificate\(\)\(Unreal\) \(Go\) \(C++\) \(C#\) \(Unreal\) auf, um den Speicherort der Datei zu .](#)

Starte eine Spielsitzung

Fügen Sie Code hinzu, um die Callback-Funktion `onStartGameSession` zu implementieren. Amazon GameLift Servers ruft diesen Callback auf, um eine Spielsitzung auf dem Serverprozess zu starten.

Die `onStartGameSession` Funktion verwendet ein [GameSession](#) Objekt als Eingabeparameter. Dieses Objekt enthält wichtige Informationen zur Spielsitzung, z. B. die maximale Spielerzahl. Es kann auch Spiel- und Spielerdaten enthalten. Die Funktionsimplementierung sollte die folgenden Aufgaben erfüllen:

- Initiieren Sie Aktionen, um basierend auf den `GameSession` Eigenschaften eine neue Spielsitzung zu erstellen. Der Spieleserver muss mindestens die Spielsitzungs-ID zuordnen, auf die Spielclients verweisen, wenn sie eine Verbindung zum Serverprozess herstellen.

- Verarbeiten Sie die Spiel- und Spielerdaten nach Bedarf. Diese Daten befinden sich im `GameSession` Objekt.
- Informiere den Amazon GameLift Servers Dienst, wenn eine neue Spielsitzung bereit ist, Spieler aufzunehmen. [Rufen Sie die Server-API-Operation `ActivateGameSession\(\)`\(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\)](#) auf. Als Reaktion auf einen erfolgreichen Aufruf ändert der Dienst den Status der Spielsitzung auf. `ACTIVE`

(Optional) Bestätigen Sie einen neuen Spieler

Wenn du den Status von Spiellersitzungen verfolgst, füge Code hinzu, um einen neuen Spieler zu validieren, wenn er sich mit einem Spieleserver verbindet. Amazon GameLift Servers verfolgt aktuelle Spieler und verfügbare Spielsitzungsplätze.

Zur Überprüfung muss ein Spielclient, der versucht, an einer Spielsitzung teilzunehmen, eine Spiellersitzungs-ID enthalten. Amazon GameLift Servers generiert diese ID, wenn dein Spiel neue Spielsitzungen startet, indem du [StartGameSessionPlacement\(\)](#) oder [StartMatchmaking\(\)](#) aufrufst. Bei diesen Anfragen wird ein offener Slot in einer Spielsitzung für die Spiellersitzung reserviert.

Wenn der Spieleserverprozess eine Verbindungsanfrage für den Spielclient erhält, ruft er `AcceptPlayerSession()` [\(C++\)](#) [\(C#\)](#) [\(Unreal\)](#) [\(Go\)](#) [\(C++\)](#) des Spielers auf. Amazon GameLift Servers überprüft daraufhin, ob die Sitzungs-ID des Spielers einem offenen Slot entspricht, der in der Spielsitzung reserviert wurde. Nach der Amazon GameLift Servers Validierung der Spiellersitzungs-ID akzeptiert der Serverprozess die Verbindung. Der Spieler kann dann an der Spielsitzung teilnehmen. Wenn die Sitzungs-ID des Spielers Amazon GameLift Servers nicht validiert wird, verweigert der Serverprozess die Verbindung.

(Optional) Melde das Ende einer Spiellersitzung

Wenn du den Status von Spiellersitzungen verfolgst, füge einen Code hinzu, um zu benachrichtigen, Amazon GameLift Servers wenn ein Spieler die Spielsitzung verlässt. Dieser Code sollte immer dann ausgeführt werden, wenn der Serverprozess eine unterbrochene Verbindung feststellt. Amazon GameLift Servers verwendet diese Benachrichtigung, um aktuelle Spieler und verfügbare Slots in der Spielsitzung zu verfolgen.

Um unterbrochene Verbindungen in Ihrem Code zu behandeln, fügen Sie der Server-API-Operation `RemovePlayerSession()` [\(C++\)](#) [\(C#\)](#) [\(Unreal\)](#) [\(Go\)](#) [\(C++\)](#) [\(C#\)](#) einen Aufruf mit der entsprechenden Spiellersitzungs-ID hinzu.

Beenden Sie eine Spielsitzung

Fügen Sie der Sequenz zum Herunterfahren des Serverprozesses Code hinzu, um zu benachrichtigen Amazon GameLift Servers, wenn eine Spielsitzung endet. Um Hosting-Ressourcen zu recyceln und zu aktualisieren, solltest du jeden Serverprozess herunterfahren, nachdem die Spielsitzung abgeschlossen ist.

[Rufen Sie zu Beginn des Codes zum Herunterfahren des Serverprozesses die Server-API-Operation `ProcessEnding\(\)` \(C++\) \(C#\) \(Unreal\) \(Go\) g \(C++\) auf](#). Dieser Aufruf benachrichtigt Amazon GameLift Servers, dass der Serverprozess heruntergefahren wird. Amazon GameLift Servers ändert den Status der Spielsitzung und den Status des Serverprozesses auf. `TERMINATED` Nach dem `ProcessEnding()` Aufrufen kann der Prozess gefahrlos heruntergefahren werden.

Reagieren Sie auf eine Benachrichtigung zum Herunterfahren des Serverprozesses

Fügen Sie Code hinzu, um den Serverprozess als Reaktion auf eine Benachrichtigung des Amazon GameLift Servers Dienstes herunterzufahren. Der Dienst sendet diese Benachrichtigung, wenn der Serverprozess ständig fehlerhaft meldet oder wenn die Instanz, auf der der Serverprozess läuft, beendet wird. Amazon GameLift Servers kann eine Instance im Rahmen einer Kapazitätsskalierung oder als Reaktion auf eine Unterbrechung der Spot-Instance beenden. Bei einer Spot-Instance-Unterbrechung wird eine Frist von zwei Minuten angezeigt, sodass der Server genügend Zeit hat, um die Verbindung der Spieler ordnungsgemäß zu trennen, die Spielstatusdaten beizubehalten und andere Säuberungsaufgaben durchzuführen.

Um eine Benachrichtigung über das Herunterfahren zu bearbeiten, nimm die folgenden Änderungen an deinem Spielservercode vor:

- [Implementieren Sie die Callback-Funktion `onProcessTerminate\(\)` \(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\)](#) Diese Funktion sollte Code aufrufen, der den Serverprozess herunterfährt.
- Rufen Sie die Server-API-Operation `GetTerminationTime()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) ([C++](#)) ([C#](#)) aus auf. If Amazon GameLift Servers hat einen Aufruf zum Stoppen des Serverprozesses ausgegeben und `GetTerminationTime()` gibt dann die geschätzte Kündigungszeit zurück.
- [Rufen Sie zu Beginn des Codes zum Herunterfahren des Spieleservers die Server-API-Operation `ProcessEnding\(\)` \(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\) auf](#). Dieser Aufruf benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess heruntergefahren wird. Der Dienst ändert dann den Serverprozessstatus auf. `TERMINATED` Nach dem `ProcessEnding()` Aufrufen kann der Prozess sicher heruntergefahren werden.

Integrieren Sie das Server-SDK für Game-Engines

Sie können den verwalteten Amazon GameLift Servers Dienst mit den meisten gängigen Spiele-Engines verwenden, die C++- oder C#-Bibliotheken unterstützen, einschließlich O3DE, Unreal Engine und Unity. Erstellen Sie die Version, die Sie für Ihr Spiel benötigen. Informationen zu Build-Anweisungen und Mindestanforderungen finden Sie in den README-Dateien jeder Version. Weitere Informationen zu verfügbaren Amazon GameLift Servers SDKs, unterstützten Entwicklungsplattformen und Betriebssystemen finden Sie unter [Holen Sie sich Amazon GameLift Servers Entwicklungstools](#) Für Spieleserver.

Zusätzlich zu den Engine-spezifischen Informationen in diesem Thema finden Sie in den folgenden Themen zusätzliche Hilfe bei der Integration von Amazon GameLift Servers mit ihren Spielservern, Clients und Services:

- [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#) – Detaillierte Anweisungen für die Integration von Amazon GameLift Servers in einen Spiel-Server.
- [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#) – Detaillierte Anleitung zur Integration in einen Spiele-Client oder -Service, einschließlich der Erstellung von Spielsitzungen und Einbindung von Spielern in Spiele.

Unreal Engine

Spiel-Server

Bereiten Sie Ihre Spieleserver für das Hosten vor, Amazon GameLift Servers indem Sie Ihrem Projekt das [Server-SDK Amazon GameLift Servers für Unreal Engine](#) hinzufügen und die erforderlichen Serverfunktionen implementieren. Hilfe beim Einrichten des Unreal-Engine-Plugins und beim Hinzufügen von Amazon GameLift Servers-Code finden Sie unter [Amazon GameLift ServersIn ein Unreal Engine-Projekt integrieren](#).

Spiele-Clients und -Services

Ermöglichen Sie den and/or Spielediensten Ihrer Spieleclients die Interaktion mit dem Amazon GameLift Servers Dienst, z. B. um verfügbare Spielsitzungen zu finden oder neue zu erstellen und Spieler zu Spielen hinzuzufügen. Die wichtigsten Client-Funktionen sind im [AWS SDK for C++](#) enthalten. Informationen zur Integration von Amazon GameLift Servers in Ihr Unreal Engine-Spielprojekt finden Sie unter [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#).

Unity

Spiel-Server

Bereiten Sie Ihre Spielservers für das Hosten vor, Amazon GameLift Servers indem Sie Ihrem Projekt das [Server-SDK Amazon GameLift Servers für C#](#) hinzufügen und die erforderlichen Serverfunktionen implementieren. Hilfe zum Einrichten mit Unity und zum Hinzufügen von Amazon GameLift Servers-Code finden Sie unter [Amazon GameLift ServersIn ein Unity-Projekt integrieren](#).

Spiele-Clients und -Services

Ermöglichen Sie den and/or Spielediensten Ihrer Spieleclients die Interaktion mit dem Amazon GameLift Servers Dienst, z. B. um verfügbare Spielsitzungen zu finden oder neue zu erstellen und Spieler zu Spielen hinzuzufügen. Die Kernclientfunktionalität wird über [AWS SDK für .NET](#) bereitgestellt. Informationen zur Integration von Amazon GameLift Servers in Ihr Unity-Spielprojekt finden Sie unter [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#).

Andere Engines

Eine vollständige Liste der für Spiele Amazon GameLift Servers SDKs verfügbaren Server und Clients finden Sie unter [the section called "Holen Sie sich Entwicklungstools"](#).

Amazon GameLift ServersIn ein Unreal Engine-Projekt integrieren

Erfahre, wie du das Amazon GameLift Servers SDK für Unreal Engine in deine Spieleprojekte integrierst, um auf den vollen Funktionsumfang des Server-SDK zuzugreifen.

Tipp

Für eine schnellere Bereitstellung probiere das Amazon GameLift Servers eigenständige Plugin für Unreal Engine aus. Es bietet geführte UI-Workflows, mit denen du deinen Spielservers schnell und mit minimalem Einrichtungsaufwand bereitstellen kannst, sodass du deine Spielkomponenten in Aktion ausprobieren kannst. Siehe [Amazon GameLift ServersPlugin für Unreal Engine](#).

Weitere Ressourcen:

- [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- [Holen Sie sich Amazon GameLift Servers Entwicklungstools](#)

Installiere das Server-SDK für Unreal

Holen Sie sich das Amazon GameLift Servers Open-Source-SDK für Unreal Engine von [GitHub](#). Die Readme-Dateien des Repositorys enthalten Voraussetzungen und Installationsanweisungen.

Richten Sie Build-Ziele und Modulregeln ein

Ändern Sie die Projektdateien Ihres Spiels, um die Build-Komponenten für die Verwendung mit Amazon GameLift Servers korrekt zu generieren.

So fügst du Client- und Server-Build-Ziele hinzu:

1. Öffne die Codedateien deines Spieleprojekts und suche die `.../Games/[your application name]Source/[your application name]Target.cs` Dateidatei. Beispiel: `.../Source/GameLiftUnrealAppTarget.cs`. (Wenn Sie Visual Studio verwenden, öffnen Sie die `.sln` Projektdatei.)
2. Kopieren Sie diese Datei, um zwei neue Zieldateien im `Source/` Verzeichnis zu erstellen.
 - Client-Ziel — Benennen Sie die neue Datei um in `[your application name]Client.Target.cs`. Bearbeiten Sie den Inhalt, um die Werte für den Klassennamen und den Zieltyp zu aktualisieren, wie im folgenden Beispielcode dargestellt:

```
using UnrealBuildTool;
using System.Collections.Generic;

public class GameLiftUnrealAppClientTarget : TargetRules
{
    public GameLiftUnrealAppClientTarget ( TargetInfo Target ) : base
    ( Target )
    {
        Type = TargetType.Client;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp" );
    }
}
```

- Serverziel — Benennen Sie die neue Datei um in `[your application name]Server.Target.cs`. Bearbeiten Sie den Inhalt, um die Werte für den Klassennamen und den Zieltyp zu aktualisieren, wie im folgenden Beispielcode dargestellt:


```
using UnrealBuildTool;
using System.Collections.Generic;

public class GameLiftUnrealAppServerTarget : TargetRules
{
    public GameLiftUnrealAppServerTarget ( TargetInfo Target ) : base
( Target )
    {
        Type = TargetType.Server;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp");
    }
}
```

3. Generieren Sie Ihre Projektdateien neu. Wenn Sie Visual Studio verwenden, können Sie mit der rechten Maustaste auf die .uproject Datei Ihres Spieleprojekts klicken und Visual Studio-Projektdateien generieren auswählen.

So aktualisieren Sie die Regeln für das Spielprojektmodul:

Aktualisiere die Modulregeln des Spielprojekts, sodass sie vom Plugin abhängig sind.

1. Öffne die Codedateien deines Spielprojekts und suche die .../Games/[*your application name*]Source/[*your application name*].Build.cs Dateidei. Beispiel: .../Source/GameLiftUnrealApp.Build.cs. (Wenn Sie Visual Studio verwenden, öffnen Sie die .sln Projektdatei.)
2. Suchen Sie die ModuleRules Klasse und aktualisieren Sie sie, wie im folgenden Beispielcode dargestellt:

```
using UnrealBuildTool;

public class GameLiftUnrealApp : ModuleRules
{
    public GameLiftUnrealApp ( ReadOnlyTargetRules Target ) : base ( Target )
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;
        PublicDependencyModuleNames.AddRange( new string[] { "Core",
"CoreUObject", "Engine", "InputCore", "HeadMountedDisplay",
"EnhancedInput" });
    }
}
```

```
// Add the following section
if (Target.Type == TargetType.Server)
{
    PublicDependencyModuleNames.Add("GameLiftServerSDK");
}
else
{
    PublicDefinitions.Add("WITH_GAMELIFT=0");
}
bEnableExceptions = true;
}
}
```

3. Nachdem Sie die neuen Zielf Dateien erstellt und die Modulregeln geändert haben, erstellen Sie Ihr Spielprojekt neu.

Füge deinem Servercode Game-Hosting-Funktionen hinzu

Nach der Installation und Einrichtung des Server-SDK besteht der nächste Schritt darin, die Game-Hosting-Funktionalität in Ihren Servercode zu integrieren. Das Server-SDK ermöglicht es deinem Spiels server, mit dem Amazon GameLift Servers Dienst zu kommunizieren, Anweisungen für Spielsitzungen zu erhalten, Status und Zustand zu melden und andere Aktionen durchzuführen.

Dieses Thema enthält Beispielcode, der die Mindestfunktionalität hinzufügt, die für das Hosten Ihres Spiels erforderlich istAmazon GameLift Servers.

Schritt 1: Aktualisiere die **GameMode** Header-Datei

1. Öffne die Codedateien deines Spielprojekts und suche die *Your-application-name*GameMode.h Dateidei. Beispiel: GameLiftUnrealAppGameMode.h. Wenn Sie Visual Studio verwenden, öffnen Sie die .sln Datei für Ihr Spieleprojekt.
2. Ändern Sie die Header-Datei so, dass sie den folgenden Beispielcode enthält. Achten Sie darauf, "GameLiftUnrealApp" durch Ihren eigenen Anwendungsnamen zu ersetzen.

Beispielcode für „h“ GameMode.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

#pragma once
```

```

#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "GameLiftUnrealAppGameMode.generated.h"

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

UCLASS(minimalapi)
class AGameLiftUnrealAppGameMode : public AGameModeBase
{
    GENERATED_BODY()

public:
    AGameLiftUnrealAppGameMode();

protected:
    virtual void BeginPlay() override;

private:
    void InitGameLift();

private:
    TSharedPtr<FProcessParameters> ProcessParameters;
};

```

Schritt 2: Füge deinem Spieleservercode die erforderlichen Server-SDK-Aufrufe hinzu

Verwende den Beispielcode in diesem Abschnitt, um deinen Gameservercode zur Verwendung mit zu integrieren Amazon GameLift Servers. Einzelheiten zur Funktionsweise des Codes finden Sie unter [Initialisieren Sie den Serverprozess](#) und [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#).

Note

Das WITH_GAMELIFT Präprozessor-Flag dient zwei Zwecken:

- Beschränkt Amazon GameLift Servers Backend-API-Aufrufe nur auf Unreal-Server-Builds
- Stellt die Kompatibilität zwischen verschiedenen Unreal-Build-Zielen sicher

1. Öffnen Sie die zugehörige *Your-application-name*GameMode.cpp Quelldatei. In unserem Beispiel:GameLiftUnrealAppGameMode.cpp.
2. Ändern Sie den Code so, dass er dem folgenden Beispielcode entspricht. Achten Sie darauf, jede Instanz von "GameLiftUnrealApp" durch Ihren eigenen Anwendungsnamen zu ersetzen.

Das bereitgestellte Codebeispiel zeigt, wie die erforderlichen Elemente für die Integration mit hinzugefügt Amazon GameLift Servers werden. Dazu zählen:

- Initialisieren Sie einen Amazon GameLift Servers API-Client.
- Implementieren Sie Callback-Funktionen, um auf Anfragen des Amazon GameLift Servers Dienstes zu antworten, einschließlich OnStartGameSessionOnProcessTerminate, und onHealthCheck
- Rufen Sie anProcessReady(), um den Amazon GameLift Servers Dienst zu benachrichtigen, wenn er bereit ist, Spielsitzungen abzuhalten.

Beispielcode GameMode für.cpp

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

#include "GameLiftUnrealAppGameMode.h"

#include "UObject/ConstructorHelpers.h"
#include "Kismet/GameplayStatics.h"

#if WITH_GAMELIFT
#include "GameLiftServerSDK.h"
#include "GameLiftServerSDKModels.h"
#endif

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

AGameLiftUnrealAppGameMode::AGameLiftUnrealAppGameMode() :
    ProcessParameters(nullptr)
{
    // Set default pawn class to our Blueprinted character
    static ConstructorHelpers::FClassFinder<APawn> PlayerPawnBPClass(TEXT("/Game/
ThirdPerson/Blueprints/BP_ThirdPersonCharacter"));
}
```

```

    if (PlayerPawnBPClass.Class != NULL)
    {
        DefaultPawnClass = PlayerPawnBPClass.Class;
    }

    UE_LOG(GameServerLog, Log, TEXT("Initializing AGameLiftUnrealAppGameMode..."));
}

void AGameLiftUnrealAppGameMode::BeginPlay()
{
    Super::BeginPlay();

#ifdef WITH_GAMELIFT
    InitGameLift();
#endif
}

void AGameLiftUnrealAppGameMode::InitGameLift()
{
#ifdef WITH_GAMELIFT
    UE_LOG(GameServerLog, Log, TEXT("Calling InitGameLift..."));

    // Getting the module first.
    FGameLiftServerSDKModule* GameLiftSdkModule =
    &FModuleManager::LoadModuleChecked<FGameLiftServerSDKModule>(FName("GameLiftServerSDK"));

    //Define the server parameters for a GameLift Anywhere fleet. These are not needed
    for a GameLift managed EC2 fleet.
    FServerParameters ServerParametersForAnywhere;

    bool bIsAnywhereActive = false;
    if (FParse::Param(FCommandLine::Get(), TEXT("glAnywhere")))
    {
        bIsAnywhereActive = true;
    }

    if (bIsAnywhereActive)
    {
        UE_LOG(GameServerLog, Log, TEXT("Configuring server parameters for
Anywhere..."));

        // If GameLift Anywhere is enabled, parse command line arguments and pass them
        in the ServerParameters object.

```

```

    FString glAnywhereWebSocketUrl = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereWebSocketUrl="),
glAnywhereWebSocketUrl))
    {
        ServerParametersForAnywhere.m_webSocketUrl =
TCHAR_TO_UTF8(*glAnywhereWebSocketUrl);
    }

    FString glAnywhereFleetId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereFleetId="),
glAnywhereFleetId))
    {
        ServerParametersForAnywhere.m_fleetId = TCHAR_TO_UTF8(*glAnywhereFleetId);
    }

    FString glAnywhereProcessId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereProcessId="),
glAnywhereProcessId))
    {
        ServerParametersForAnywhere.m_processId =
TCHAR_TO_UTF8(*glAnywhereProcessId);
    }
    else
    {
        // If no ProcessId is passed as a command line argument, generate a
randomized unique string.
        FString TimeString = FString::FromInt(std::time(nullptr));
        FString ProcessId = "ProcessId_" + TimeString;
        ServerParametersForAnywhere.m_processId = TCHAR_TO_UTF8(*ProcessId);
    }

    FString glAnywhereHostId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereHostId="),
glAnywhereHostId))
    {
        ServerParametersForAnywhere.m_hostId = TCHAR_TO_UTF8(*glAnywhereHostId);
    }

    FString glAnywhereAuthToken = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAuthToken="),
glAnywhereAuthToken))
    {
        ServerParametersForAnywhere.m_authToken =
TCHAR_TO_UTF8(*glAnywhereAuthToken);
    }

```

```

    }

    FString glAnywhereAwsRegion = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAwsRegion="),
glAnywhereAwsRegion))
    {
        ServerParametersForAnywhere.m_awsRegion =
TCHAR_TO_UTF8(*glAnywhereAwsRegion);
    }

    FString glAnywhereAccessKey = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAccessKey="),
glAnywhereAccessKey))
    {
        ServerParametersForAnywhere.m_accessKey =
TCHAR_TO_UTF8(*glAnywhereAccessKey);
    }

    FString glAnywhereSecretKey = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereSecretKey="),
glAnywhereSecretKey))
    {
        ServerParametersForAnywhere.m_secretKey =
TCHAR_TO_UTF8(*glAnywhereSecretKey);
    }

    FString glAnywhereSessionToken = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereSessionToken="),
glAnywhereSessionToken))
    {
        ServerParametersForAnywhere.m_sessionToken =
TCHAR_TO_UTF8(*glAnywhereSessionToken);
    }

    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_YELLOW);
    UE_LOG(GameServerLog, Log, TEXT(">>>> WebSocket URL: %s"),
*ServerParametersForAnywhere.m_webSocketUrl);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Fleet ID: %s"),
*ServerParametersForAnywhere.m_fleetId);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Process ID: %s"),
*ServerParametersForAnywhere.m_processId);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Host ID (Compute Name): %s"),
*ServerParametersForAnywhere.m_hostId);

```

```

        UE_LOG(GameServerLog, Log, TEXT(">>>> Auth Token: %s"),
*ServerParametersForAnywhere.m_authToken);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Aws Region: %s"),
*ServerParametersForAnywhere.m_awsRegion);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Access Key: %s"),
*ServerParametersForAnywhere.m_accessKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Secret Key: %s"),
*ServerParametersForAnywhere.m_secretKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Session Token: %s"),
*ServerParametersForAnywhere.m_sessionToken);
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }

    UE_LOG(GameServerLog, Log, TEXT("Initializing the GameLift Server..."));

    //InitSDK will establish a local connection with GameLift's agent to enable further
communication.
    FGameLiftGenericOutcome InitSdkOutcome = GameLiftSdkModule-
>InitSDK(ServerParametersForAnywhere);
    if (InitSdkOutcome.IsSuccess())
    {
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
        UE_LOG(GameServerLog, Log, TEXT("GameLift InitSDK succeeded!"));
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }
    else
    {
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
        UE_LOG(GameServerLog, Log, TEXT("ERROR: InitSDK failed : ("));
        FGameLiftError GameLiftError = InitSdkOutcome.GetError();
        UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"), *GameLiftError.m_errorMessage);
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
        return;
    }

    ProcessParameters = MakeShared<FProcessParameters>();

    //When a game session is created, Amazon GameLift Servers sends an activation
request to the game server and passes along the game session object containing game
properties and other settings.
    //Here is where a game server should take action based on the game session object.
    //Once the game server is ready to receive incoming player connections, it should
invoke GameLiftServerAPI.ActivateGameSession()

```



```

    ProcessParameters->OnStartGameSession.BindLambda( [=]
(Aws::GameLift::Server::Model::GameSession InGameSession)
    {
        FString GameSessionId = FString(InGameSession.GetGameSessionId());
        UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"),
*GameSessionId);
        GameLiftSdkModule->ActivateGameSession();
    });

    //OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback
before shutting down an instance hosting this game server.
    //It gives this game server a chance to save its state, communicate with services,
etc., before being shut down.
    //In this case, we simply tell Amazon GameLift Servers we are indeed going to
shutdown.
    ProcessParameters->OnTerminate.BindLambda( [=]()
    {
        UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
        // First call ProcessEnding()
        FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule-
>ProcessEnding();
        // Then call Destroy() to free the SDK from memory
        FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();
        // Exit the process with success or failure
        if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess()) {
            UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
        }
        else {
            if (!processEndingOutcome.IsSuccess()) {
                const FGameLiftError& error = processEndingOutcome.GetError();
                UE_LOG(GameServerLog, Error, TEXT("ProcessEnding() failed. Error:
%s"),
                    error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
*error.m_errorMessage);
            }
            if (!destroyOutcome.IsSuccess()) {
                const FGameLiftError& error = destroyOutcome.GetError();
                UE_LOG(GameServerLog, Error, TEXT("Destroy() failed. Error: %s"),
                    error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
*error.m_errorMessage);
            }
        }
    });

```

```

//This is the HealthCheck callback.
//Amazon GameLift Servers will invoke this callback every 60 seconds or so.
//Here, a game server might want to check the health of dependencies and such.
//Simply return true if healthy, false otherwise.
//The game server has 60 seconds to respond with its health status. Amazon GameLift
Servers will default to 'false' if the game server doesn't respond in time.
//In this case, we're always healthy!
ProcessParameters->OnHealthCheck.BindLambda([]()
{
    UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));
    return true;
});

//GameServer.exe -port=7777 LOG=server.mylog
ProcessParameters->port = FURL::UrlConfig.DefaultPort;
TArray<FString> CommandLineTokens;
TArray<FString> CommandLineSwitches;

FCommandLine::Parse(FCommandLine::Get(), CommandLineTokens, CommandLineSwitches);

for (FString SwitchStr : CommandLineSwitches)
{
    FString Key;
    FString Value;

    if (SwitchStr.Split("=", &Key, &Value))
    {
        if (Key.Equals("port"))
        {
            ProcessParameters->port = FString::Atoi(*Value);
        }
    }
}

//Here, the game server tells Amazon GameLift Servers where to find game session
log files.
//At the end of a game session, Amazon GameLift Servers uploads everything in the
specified
//location and stores it in the cloud for access later.
TArray<FString> Logfiles;
Logfiles.Add(TEXT("GameLiftUnrealApp/Saved/Logs/server.log"));
ProcessParameters->logParameters = Logfiles;

```

```
//The game server calls ProcessReady() to tell Amazon GameLift Servers it's ready
to host game sessions.
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready..."));
FGameLiftGenericOutcome ProcessReadyOutcome = GameLiftSdkModule-
>ProcessReady(*ProcessParameters);

if (ProcessReadyOutcome.IsSuccess())
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
    UE_LOG(GameServerLog, Log, TEXT("Process Ready!"));
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}
else
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
    UE_LOG(GameServerLog, Log, TEXT("ERROR: Process Ready Failed!"));
    FGameLiftError ProcessReadyError = ProcessReadyOutcome.GetError();
    UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"),
*ProcessReadyError.m_errorMessage);
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}

UE_LOG(GameServerLog, Log, TEXT("InitGameLift completed!"));
#endif
}
```

Schritt 3: Baue das Spielprojekt neu

- Erstellen Sie ein Spielprojekt für die beiden folgenden Zieltypen: Entwicklungseditor und Entwicklungsserver.

Note

Sie müssen die Lösung nicht neu erstellen. Erstellen Sie das Projekt stattdessen unter dem /Games/ Ordner für Ihre App. Andernfalls erstellt Visual Studio das gesamte UE5 Projekt neu, was bis zu einer Stunde dauern kann.

Paketiere deinen Gameserver für das Hosting

Da dein Gameserver-Code jetzt mit der mindestens erforderlichen Server-SDK-Funktionalität integriert ist, bist du bereit, deinen Gameserver-Build mit dem Unreal Editor zu verpacken.

Um den Gameserver-Build zu verpacken

1. Öffne das Spielprojekt im Unreal Editor.
2. Folge den Schritten im Unreal Editor, um deinen Gameserver zu packen:
 - Wähle deine Zielplattform (Windows oder Linux).
 - Wählen Sie Ihr Server-Build-Ziel aus (*[your application name]*Server).

Der Paketierungsprozess generiert die ausführbare Datei Ihres Spieleservers: *[your application name]*Server.exe.

3. Bereiten Sie Ihren Spieleserver-Build für die Bereitstellung auf Hosting-Ressourcen vor. Der Build sollte die folgenden Dateien enthalten:
 - Die ausführbare Datei deines Gameservers
 - Wenn du Unreal Engine Version 5.5 oder älter verwendest, füge die folgenden Dateien für Windows-Builds hinzu. Sie finden sie in Ihrer Quellversion der Unreal Engine:
 - VC_redist.x64.exe (UnrealEngine\Engine\Source\Programs\PrereqInstaller\Resources\VCRedist\)
 - UEPrereqSetup_x64.exe or UE5PrereqSetup_x64.exe (UnrealEngine\Engine\Extras\Redist\en-us\)
 - Alle anderen erforderlichen Abhängigkeiten für deinen Spieleserver.
 - OpenSSL-Bibliotheken, falls erforderlich. Du kannst diesen Schritt überspringen, wenn dein Spieleserver in das Amazon GameLift Servers Server-SDK Version 5.3 oder höher integriert ist. [Die neueste Server-SDK-Version ist hier verfügbar.](#)

OpenSSL-Bibliotheken sind für Spieleserver erforderlich, die mit dem Server-SDK für Unreal, Versionen 5.2 und früher, erstellt wurden

Sie müssen dieselbe Version der OpenSSL-Bibliotheken einbinden, die beim Packen des Spieleservers in Unreal verwendet wurden. Diese Bibliotheken befinden sich in deiner Game-Engine-Quelle. Der Speicherort hängt von Ihrer Entwicklungsumgebung ab:

Unter Windows:

- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libimobiledevice\x64\libssl-1_1-x64.dll

- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libmobiledevice\x64\libcrypto-1_1-x64.dll

Unter Linux:

- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libssl.so.1.1
- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libcrypto.so.1.1

Kopiere die OpenSSL-Bibliotheken in dein Game-Build-Paket im selben Verzeichnis wie die ausführbare Datei des Spielservers.

Nächste Schritte

Nachdem Sie nun einen Gameserver-Build mit den mindestens erforderlichen Funktionen für das Hosten vorbereitet haben Amazon GameLift Servers, sollten Sie die folgenden möglichen nächsten Schritte in Betracht ziehen:

- Stellen Sie Ihren integrierten Spielservers für Tests und Entwicklungen bereit. Mit einer Anywhere-Flotte kannst du deinen lokalen Computer als Hosting-Ressource einrichten und damit deine Gameserver- und Spielclient-Verbindungen testen. Stellen Sie für Cloud-basiertes Hosting Ihren Spielservers auf einer verwalteten EC2- oder verwalteten Containerflotte bereit. Anleitungen finden Sie in den folgenden Themen:
 - [Für iterative Entwicklung mit Amazon GameLift Servers Anywhere einrichten](#)
 - [Amazon GameLift Servers Überall Flotten](#)
 - [Amazon GameLift Servers verwaltete EC2 Flotten](#)
 - [Amazon GameLift Servers verwaltete Containerflotten](#)
- Passe deine Gameserver-Integration an, indem du optionale Funktionen hinzufügst. Vielleicht möchtest du beispielsweise Spielsitzungen mit einem bestimmten Spieler hinzufügen IDs, Matchmaking-Backfill einrichten oder den Spielserverszugriff auf deine anderen AWS Ressourcen (wie eine Datenbank oder einen Inhaltsspeicherdienst) verwalten. Anleitungen finden Sie in den folgenden Themen:
 - [Füge Amazon GameLift Servers es deinem Spielservers mit dem Server-SDK hinzu](#)
 - [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- Passe deine Spielclient-Komponente an, um Spielsitzungen anzufordern, Verbindungsinformationen zu erhalten und eine direkte Verbindung zu einem Spielservers herzustellen, um ein Spiel zu spielen. Anleitungen finden Sie in den folgenden Themen:

- [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#)

Amazon GameLift ServersIn ein Unity-Projekt integrieren

Erfahren Sie, wie Sie das Amazon GameLift Servers SDK for Unity in Ihre Spieleprojekte integrieren können, um auf den vollständigen Funktionsumfang des Server-SDK zuzugreifen.

Tipp

Für eine schnellere Bereitstellung probieren Sie das Amazon GameLift Servers eigenständige Plugin für Unity aus. Es bietet geführte Benutzeroberflächenworkflows zur schnellen Bereitstellung Ihres Spielservers mit minimalem Einrichtungsaufwand, sodass Sie Ihre Spielkomponenten in Aktion ausprobieren können. Siehe [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#).

Weitere Ressourcen:

- [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- [the section called “Holen Sie sich Entwicklungstools”](#)

Installieren Sie das Server-SDK SDK for Unity

Holen Sie sich die Open-Source-Version Amazon GameLift Servers für Unity von [GitHub](#). Die Readme-Dateien des Repositorys enthalten Voraussetzungen und Installationsanweisungen.

Richten Sie eine Amazon GameLift Servers Anywhere-Flotte zum Testen ein

Sie können Ihre Entwicklungs-Workstation als Amazon GameLift Servers Anywhere-Hosting-Flotte einrichten, um Ihre Amazon GameLift Servers Integration iterativ zu testen. Mit dieser Konfiguration können Sie Spielserverprozesse auf Ihrer Workstation starten, Spielerbeitritts- oder Matchmaking-Anfragen senden, Amazon GameLift Servers um Spielsitzungen zu starten, und Clients mit den neuen Spielsitzungen verbinden. Mit deiner eigenen Workstation, die als Hosting-Server eingerichtet ist, kannst du alle Aspekte deiner Spieleintegration mit Amazon GameLift Servers überwachen.

Anweisungen zur Einrichtung deiner Workstation findest du unter [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#) So führen Sie die folgenden Schritte durch:

1. Erstellen Sie einen benutzerdefinierten Speicherort für Ihre Workstation.

2. Erstellen Sie eine Amazon GameLift Servers Anywhere-Flotte mit Ihrem neuen benutzerdefinierten Standort. Bei Erfolg gibt diese Anfrage eine Flotten-ID zurück. Notieren Sie sich diesen Wert, da Sie ihn später benötigen werden.
3. Registrieren Sie Ihre Workstation als Computer in der neuen Anywhere-Flotte. Geben Sie einen eindeutigen Computernamen und die IP-Adresse für Ihre Workstation an. Bei Erfolg gibt diese Anfrage einen Service SDK-Endpunkt in Form einer WebSocket URL zurück. Notieren Sie sich diesen Wert, da Sie ihn später benötigen werden.
4. Generieren Sie ein Authentifizierungstoken für Ihren Workstation-Computer. Diese kurzlebige Authentifizierung umfasst das Token und ein Ablaufdatum. Ihr Spieleserver verwendet es, um die Kommunikation mit dem Amazon GameLift Servers Dienst zu authentifizieren. Speichere die Authentifizierung auf deinem Workstation-Computer, damit deine laufenden Gameserver-Prozesse darauf zugreifen können.

Fügen Sie Amazon GameLift Servers Servercode zu Ihrem Unity-Projekt hinzu

Ihr Spieleserver kommuniziert mit dem Amazon GameLift Servers Dienst, um Anweisungen zu erhalten und den aktuellen Status zu melden. Um dies zu erreichen, fügst du Gameservercode hinzu, der das Amazon GameLift Servers Server-SDK verwendet.

Das bereitgestellte Codebeispiel veranschaulicht die grundlegenden erforderlichen Integrationselemente. Es verwendet `aMonoBehavior`, um eine einfache Initialisierung des Spieleservers mit Amazon GameLift Servers zu veranschaulichen. Das Beispiel geht davon aus, dass der Spieleserver zu Testzwecken auf einer Amazon GameLift Servers Anywhere-Flotte läuft. Es enthält Code für:

- Initialisieren Sie einen Amazon GameLift Servers API-Client. Das Beispiel verwendet die Version von `InitSDK()` mit Serverparametern für Ihre Anywhere-Flotte und Rechenleistung. Verwenden Sie die WebSocket URL, die Flotten-ID, den Computernamen (Host-ID) und das Authentifizierungstoken, wie im vorherigen Thema definiert [Richten Sie eine Amazon GameLift Servers Anywhere-Flotte zum Testen ein](#).
- Implementieren Sie Rückruffunktionen, um auf Anfragen des Amazon GameLift Servers Dienstes zu antworten, einschließlich `OnStartGameSession`, `OnProcessTerminate`, und `onHealthCheck`.
- Rufen Sie `ProcessReady()` mit einem bestimmten Port auf, um den Amazon GameLift Servers Dienst zu benachrichtigen, wenn der Prozess bereit ist, Spielsitzungen abzuhalten.

Der bereitgestellte Beispielcode stellt die Kommunikation mit dem Amazon GameLift Servers Dienst her. Es implementiert auch eine Reihe von Rückruffunktionen, die auf Anfragen des Amazon GameLift Servers Dienstes reagieren. Weitere Informationen zu den einzelnen Funktionen und zur Funktionsweise des Codes finden Sie unter [Initialisieren des Serverprozesses](#). Weitere Informationen zu den SDK-Aktionen und Datentypen, die in diesem Code verwendet werden, finden Sie [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#) unter.

Der Beispielcode zeigt, wie Sie die erforderlichen Funktionen hinzufügen, wie unter [Amazon GameLift ServersZu Ihrem Spieleserver hinzufügen](#) beschrieben. Weitere Informationen zu Server-SDK-Aktionen finden Sie unter [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#).

Beispielcode für die Integration

```
using System.Collections.Generic;
using Aws.GameLift.Server;
using UnityEngine;

public class ServerSDKManualTest : MonoBehaviour
{
    //This example is a simple integration that initializes a game server process
    //that is running on an Amazon GameLift Servers Anywhere fleet.
    void Start()
    {
        //Identify port number (hard coded here for simplicity) the game server is
        //listening on for player connections
        var listeningPort = 7777;

        //WebSocketUrl from RegisterHost call
        var websocketUrl = "wss://us-west-2.api.amazongamelift.com";

        //Unique identifier for this process
        var processId = "myProcess";

        //Unique identifier for your host that this process belongs to
        var hostId = "myHost";

        //Unique identifier for your fleet that this host belongs to
        var fleetId = "myFleet";

        //Authorization token for this host process
        var authToken = "myAuthToken";
    }
}
```



```

//Server parameters are required for an Amazon GameLift Servers Anywhere fleet.
//They are not required for an Amazon GameLift Servers managed EC2 fleet.
ServerParameters serverParameters = new ServerParameters(
    webSocketUrl,
    processId,
    hostId,
    fleetId,
    authToken);

//InitSDK establishes a local connection with an Amazon GameLift Servers agent
//to enable further communication.
var initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
if (initSDKOutcome.Success)
{
    //Implement callback functions
    ProcessParameters processParameters = new ProcessParameters(
        //Implement OnStartGameSession callback
        (gameSession) => {
            //Amazon GameLift Servers sends a game session activation request
            to the game server
            //with game session object containing game properties and other
            settings.
            //Here is where a game server takes action based on the game
            session object.
            //When the game server is ready to receive incoming player
            connections,
            //it invokes the server SDK call ActivateGameSession().
            GameLiftServerAPI.ActivateGameSession();
        },
        (updateGameSession) => {
            //Amazon GameLift Servers sends a request when a game session is
            updated (such as for
            //FlexMatch backfill) with an updated game session object.
            //The game server can examine matchmakerData and handle new
            incoming players.
            //updateReason explains the purpose of the update.
        },
        () => {
            //Implement callback function OnProcessTerminate
            //Amazon GameLift Servers invokes this callback before shutting
            down the instance hosting this game server.
            //It gives the game server a chance to save its state, communicate
            with services, etc.,

```

```

        //and initiate shut down. When the game server is ready to shut
down, it invokes the
        //server SDK call ProcessEnding() to tell Amazon GameLift Servers
it is shutting down.
        GameLiftServerAPI.ProcessEnding();
    },
    () => {
        //Implement callback function OnHealthCheck
        //Amazon GameLift Servers invokes this callback approximately every
60 seconds.
        //A game server might want to check the health of dependencies,
etc.
        //Then it returns health status true if healthy, false otherwise.
        //The game server must respond within 60 seconds, or Amazon
GameLift Servers records 'false'.
        //In this example, the game server always reports healthy.
        return true;
    },
    //The game server gets ready to report that it is ready to host game
sessions
    //and that it will listen on port 7777 for incoming player connections.
    listeningPort,
    new LogParameters(new List<string>()
    {
        //Here, the game server tells Amazon GameLift Servers where to find
game session log files.
        //At the end of a game session, Amazon GameLift Servers uploads
everything in the specified
        //location and stores it in the cloud for access later.
        "/local/game/logs/myserver.log"
    }));

    //The game server calls ProcessReady() to tell Amazon GameLift Servers it's
ready to host game sessions.
    var processReadyOutcome =
GameLiftServerAPI.ProcessReady(processParameters);
    if (processReadyOutcome.Success)
    {
        print("ProcessReady success.");
    }
    else
    {
        print("ProcessReady failure : " +
processReadyOutcome.Error.ToString());
    }
}

```

```
        }
    }
    else
    {
        print("InitSDK failure : " + initSDKOutcome.Error.ToString());
    }
}

void OnApplicationQuit()
{
    //Make sure to call GameLiftServerAPI.ProcessEnding() and
    GameLiftServerAPI.Destroy() before terminating the server process.
    //These actions notify Amazon GameLift Servers that the process is terminating
    and frees the API client from memory.
    GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();
    GameLiftServerAPI.Destroy();
    if (processEndingOutcome.Success)
    {
        Environment.Exit(0);
    }
    else
    {
        Console.WriteLine("ProcessEnding() failed. Error: " +
        processEndingOutcome.Error.ToString());
        Environment.Exit(-1);
    }
}
}
```

Nächste Schritte

Nachdem Sie nun einen Spieleserver-Build mit den mindestens erforderlichen Funktionen für das Hosten vorbereitet haben Amazon GameLift Servers, sollten Sie die folgenden möglichen nächsten Schritte in Betracht ziehen:

- Stellen Sie Ihren integrierten Spieleserver für Tests und Entwicklungen bereit. Mit einer Anywhere-Flotte kannst du deinen lokalen Computer als Hosting-Ressource einrichten und damit deine Gameserver- und Spielclient-Verbindungen testen. Stellen Sie für Cloud-basiertes Hosting Ihren Spieleserver auf einer verwalteten EC2- oder verwalteten Containerflotte bereit. Anleitungen finden Sie in den folgenden Themen:
 - [Für iterative Entwicklung mit Amazon GameLift Servers Anywhere einrichten](#)
 - [Amazon GameLift Servers Überall Flotten](#)

- [Amazon GameLift Serversverwaltete EC2 Flotten](#)
- [Amazon GameLift Serversverwaltete Containerflotten](#)
- Passe deine Gameserver-Integration an, indem du optionale Funktionen hinzufügst. Vielleicht möchtest du beispielsweise Spielsitzungen mit einem bestimmten Spieler hinzufügen IDs, Matchmaking-Backfill einrichten oder den Spielserverzugriff auf deine anderen AWS Ressourcen (wie eine Datenbank oder einen Inhaltsspeicherdienst) verwalten. Anleitungen finden Sie in den folgenden Themen:
 - [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#)
 - [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- Passe deine Spielclient-Komponente an, um Spielsitzungen anzufordern, Verbindungsinformationen zu erhalten und eine direkte Verbindung zu einem Spieleserver herzustellen, um ein Spiel zu spielen. Anleitungen finden Sie in den folgenden Themen:
 - [Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität](#)

Einen Gameserver-Build für die Bereitstellung verpacken

Bereiten Sie Ihre Gameserver-Software für den Einsatz auf Ihren Hosting-Ressourcen vor. Nach der Bereitstellung wird die Software auf jeder Hosting-Ressource installiert. Anschließend werden ein oder mehrere Spielserverprozesse gestartet und bereit gemacht, Spielsitzungen für Spieler zu hosten.

Die Vorbereitung eines Spieleservers hängt von der Art der verwendeten Amazon GameLift Servers Hosting-Optionen ab. Alle Spielserver-Builds müssen in das Server-SDK für integriert seinAmazon GameLift Servers, wie unter beschrieben[Integrieren Sie einen Spieleserver mit Amazon GameLift Servers](#).

Die Themen in diesem Abschnitt enthalten Anleitungen dazu, wie Sie Ihre Software für den Einsatz in den folgenden Szenarien vorbereiten können.

- Für verwaltetes EC2 Hosting packen Sie Ihre Serversoftware und laden Sie sie Amazon GameLift Servers zur Bereitstellung auf hoch.
- Erstellen Sie für das Hosting von verwalteten Containern ein Container-Image mit Ihrer Serversoftware und speichern Sie es zur Bereitstellung in Amazon Elastic Container Registry.
- Für das Hosting mit Amazon GameLift Servers Anywhere packen Sie Ihre Serversoftware nach Bedarf für die Installation auf Ihren eigenen Hosting-Ressourcen.

 Note

Wenn Sie ein Amazon GameLift Servers Realtime konfiguriertes Skript bereitstellen, finden Sie weitere Informationen unter [Ein Skript hochladen für Amazon GameLift ServersRealtime](#).

Themen

- [Erstelle einen Gameserver-Build für Amazon GameLift Servers](#)
- [Erstellen Sie ein Container-Image für Amazon GameLift Servers](#)

Erstelle einen Gameserver-Build für Amazon GameLift Servers

Nachdem Sie Ihren Spieleserver in Amazon GameLift Servers (siehe [Bereite ein Spiel für das Hosten vor mit Amazon GameLift Servers](#)) integriert haben, installieren Sie die Gameserver-Software zum Hosten auf Ihren Rechenressourcen. Dieser Vorgang hängt von der Art des Amazon GameLift Servers Hostings ab, das Sie verwenden.

Für verwaltetes Hosting bereitstellen

Wenn Sie Amazon GameLift Servers verwaltetes EC2-Hosting verwenden, müssen Sie Ihre Gameserver-Software packen und auf Amazon GameLift Servers hochladen. Wenn Sie eine verwaltete Flotte erstellen, Amazon GameLift Servers wird diese automatisch auf jeder Flotteninstanz bereitgestellt.

In den Themen in diesem Abschnitt wird beschrieben, wie Sie Ihre Build-Dateien für den Upload verpacken, ein optionales Build-Installationsskript erstellen und dann die Dateien mit dem [AWS Command Line Interface \(AWS CLI\)](#) oder AWS SDK hochladen.

Bereitstellen für Anywhere-Hosting

Wenn Sie Amazon GameLift Servers Anywhere-Flotten für selbstverwaltetes Hosting verwenden, liegt es in Ihrer Verantwortung, die Gameserver-Software auf jedem Computer in einer Flotte zu installieren und sie auf dem neuesten Stand zu halten.

Wenn ein integrierter Spieleserver-Prozess gestartet wird, wird er automatisch initialisiert und stellt die Kommunikation mit dem Dienst her. Amazon GameLift Servers Der Serverprozess startet Spielsitzungen anhand von Anweisungen von Amazon GameLift Servers und meldet Aktivitäten an den Dienst zurück.

Themen

- [Packen Ihrer Spielserver-Build-Dateien](#)
- [Hinzufügen eines Build-Installationsskripts](#)
- [Erstellen Sie eine Amazon GameLift Servers Build-Ressource für Managed Hosting](#)

Packen Ihrer Spielserver-Build-Dateien

Bevor du deinen konfigurierten Spieleserver auf hochlädst Amazon GameLift Servers, packe die Spiele-Build-Dateien in ein Build-Verzeichnis. Dieser Prozess ist erforderlich, wenn Sie mit einer verwalteten EC2-Flotte hosten, und ist eine bewährte Methode, wenn Sie mit einer Anywhere-Flotte hosten. Das Build-Verzeichnis sollte alle Komponenten enthalten, die für den Betrieb Ihrer Spieleserver und das Hosten von Spielsitzungen erforderlich sind. Dies könnte Folgendes beinhalten:

- Spieleserver-Binärdateien — Die Binärdateien, die für den Betrieb des Spieleservers erforderlich sind. Ein Build kann Binärdateien für mehrere Spieleserver enthalten, die für die Ausführung auf derselben Plattform erstellt wurden. Eine Liste der unterstützten Plattformen finden Sie unter [Holen Sie sich Amazon GameLift Servers Entwicklungstools](#).
- Abhängigkeiten — Alle abhängigen Dateien, die die ausführbaren Dateien deines Gameservers zum Ausführen benötigen. Beispiele sind u. a. Assets, Konfigurationsdateien und abhängige Bibliotheken.

Note

Fügen Sie bei Spiele-Builds, die mit dem Server-SDK Amazon GameLift Servers für C++ erstellt wurden (einschließlich solcher, die mit dem Unreal-Plugin erstellt wurden), die OpenSSL-DLL für dieselbe Version von OpenSSL hinzu, mit der Sie das Server-SDK erstellt haben. Weitere Informationen finden Sie in der README-Datei des Server-SDK.

- Installationsskript (optional) — Eine Skriptdatei für Aufgaben zur Installation deines Spiel-Builds auf Amazon GameLift Servers Hosting-Servern. Platziere diese Datei im Stammverzeichnis des Build-Verzeichnisses. Amazon GameLift Servers führt das Installationsskript als Teil der Flottenerstellung aus.

Sie können jede Anwendung in Ihrem Build einrichten, einschließlich Ihres Installationsskripts, um sicher auf Ihre Ressourcen in anderen AWS Diensten zuzugreifen. Informationen dazu, wie Sie dies

tun können, finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).

Nachdem du deine Build-Dateien gepackt hast, vergewissere dich, dass dein Spieleserver auf einer Neuinstallation deines Zielbetriebssystems ausgeführt werden kann, um sicherzustellen, dass alle erforderlichen Abhängigkeiten enthalten sind und dass dein Installationsskript korrekt ist.

Hinzufügen eines Build-Installationsskripts

Erstelle ein Installationsskript für das Betriebssystem (OS) deines Spiele-Builds:

- Windows: Erstelle eine Batch-Datei mit dem Namen `install.bat`.
- Linux: Erstellen Sie eine Shell-Skriptdatei mit dem Namen `install.sh`.

Beachten Sie beim Erstellen eines Installationsskripts Folgendes:

- Das Skript kann keine Benutzereingaben annehmen.
- Amazon GameLift Servers installiert den Build und erstellt die Dateiverzeichnisse in Ihrem Build-Paket auf einem Hosting-Server an den folgenden Speicherorten neu:
 - Windows-Flotten: `C:\game`
 - Linux-Flotten: `/local/game`
- Während des Installationsvorgangs für Linux-Flotten hat der Run-as-Benutzer eingeschränkten Zugriff auf die Instanzdateistruktur. Dieser Benutzer hat volle Rechte für das Verzeichnis, in dem Ihre Build-Dateien installiert sind. Wenn Ihr Installationsskript Aktionen ausführt, für die Administratorrechte erforderlich sind, geben Sie den Administratorzugriff mit `ansudo`. Der Run-as-Benutzer für Windows-Flotten verfügt standardmäßig über Administratorrechte. Fehlende Berechtigungen für das Installationsskript generieren eine Ereignismeldung, die auf ein Problem mit dem Skript hinweist.
- Amazon GameLift Servers unterstützt unter Linux gängige Shell-Interpreter-Sprachen wie Bash. Fügen Sie einen shebang (z. B. `#!/bin/bash`) oben in Ihrem Installationsskript ein. Um die Unterstützung Ihrer bevorzugten Shell-Befehle zu überprüfen, greifen Sie remote auf eine aktive Linux-Instance zu und öffnen Sie eine Shell-Eingabeaufforderung. Weitere Informationen finden Sie unter [Stellen Sie eine Remoteverbindung zu Amazon GameLift Servers Flotteninstanzen her](#).
- Das Installationsskript kann sich nicht auf eine VPC-Peering-Verbindung verlassen. Eine VPC-Peering-Verbindung ist erst nach der Amazon GameLift Servers Installation der Build-On-Flotteninstanzen verfügbar.

Example Windows installiert die Bash-Datei

Diese `install.bat` Beispieldatei installiert die für den Spieleserver erforderlichen Visual C++-Laufzeitkomponenten und schreibt die Ergebnisse in eine Protokolldatei. Das Skript enthält die Komponentendatei im Build-Paket im Stammverzeichnis.

```
vcxredist_x64.exe /install /quiet /norestart /log c:\game\vcxredist_2013_x64.log
```

Example Linux-Shell-Skript installieren

Diese `install.sh` Beispieldatei verwendet Bash im Installationsskript und schreibt die Ergebnisse in eine Protokolldatei.

```
#!/bin/bash
echo 'Hello World' > install.log
```

Diese `install.sh` Beispieldatei zeigt, wie Sie den CloudWatch Amazon-Agenten verwenden können, um Metriken auf Systemebene und benutzerdefinierte Metriken zu sammeln und die Protokollrotation zu handhaben. Da in einer Service-VPC Amazon GameLift Servers ausgeführt wird, müssen Sie Amazon GameLift Servers Berechtigungen erteilen, um in Ihrem Namen eine AWS Identity and Access Management (IAM-) Rolle anzunehmen. Um die Übernahme einer Rolle Amazon GameLift Servers zu ermöglichen, erstellen Sie eine Rolle, die die AWS verwaltete Richtlinie `CloudWatchAgentAdminPolicy`, und verwenden Sie diese Rolle, wenn Sie eine Flotte erstellen.

```
sudo yum install -y amazon-cloudwatch-agent
sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
sudo yum install -y collectd
cat <<'EOF' > /tmp/config.json
{
  "agent": {
    "metrics_collection_interval": 60,
    "run_as_user": "root",
    "credentials": {
      "role_arn": "arn:aws:iam::account:role/rolename"
    }
  },
  "logs": {
    "logs_collected": {
      "files": {
```



```

        "collect_list": [
            {
                "file_path": "/tmp/log",
                "log_group_name": "gllog",
                "log_stream_name": "{instance_id}"
            }
        ]
    },
    "metrics": {
        "namespace": "GL_Metric",
        "append_dimensions": {
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}"
        },
        "metrics_collected": {
            // Configure metrics you want to collect.
            // For more information, see Manually create or edit the CloudWatch agent configuration file.
        }
    }
}
EOF
sudo mv /tmp/config.json /opt/aws/amazon-cloudwatch-agent/bin/config.json
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
sudo systemctl enable amazon-cloudwatch-agent.service

```

Erstellen Sie eine Amazon GameLift Servers Build-Ressource für Managed Hosting

Wenn Sie einen Build erstellen und Ihre Dateien hochladen, haben Sie zwei Optionen:

- [Erstellen Sie einen Build aus einem Dateiverzeichnis](#). Dies ist die einfachste und am häufigsten verwendete Option.
- [Erstellen Sie einen Build mit Dateien in Amazon Simple Storage Service \(Amazon S3\)](#). Mit dieser Option können Sie Ihre Build-Versionen in Amazon S3 verwalten.

Amazon GameLift ServersErstellt mit beiden Methoden eine neue Build-Ressource mit einer eindeutigen Build-ID und anderen Metadaten. Der Build startet im Status Initialisiert. Nachdem die

Amazon GameLift Servers Spielserverdateien abgerufen wurden, wechselt der Build in den Status Bereit.

Wenn der Build fertig ist, kannst du ihn in einer neuen Amazon GameLift Servers Flotte einsetzen. Weitere Informationen finden Sie unter [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#). Wenn Amazon GameLift Servers die neue Flotte eingerichtet wird, werden die Build-Dateien auf jede Flotteninstanz heruntergeladen und die Build-Dateien installiert.

Erstellen Sie einen Build aus einem Dateiverzeichnis

Verwenden Sie den [upload-build](#) AWS CLI Befehl, um einen Spiel-Build zu erstellen, der an einem beliebigen Ort, einschließlich eines lokalen Verzeichnisses, gespeichert ist. Dieser Befehl erstellt einen neuen Build-Datensatz in Amazon GameLift Servers und lädt Dateien von einem von Ihnen angegebenen Ort hoch.

Senden Sie eine Upload-Anfrage. Geben Sie in einem Befehlszeilenfenster den folgenden upload-build Befehl und die folgenden Parameter ein.

```
aws gamelift upload-build \  
  --name user-defined name of build \  
  --operating-system supported OS \  
  --server-sdk-version server SDK for Amazon GameLift Servers version \  
  --build-root build path \  
  --build-version user-defined build number \  
  --region region name
```

- **operating-system**— Die Laufzeitumgebung des Gameserver-Builds. Sie müssen einen Betriebssystemwert angeben. Sie können dies später nicht aktualisieren.
- **server-sdk-version**— Die Version des Amazon GameLift Servers Server-SDK, in die dein Spieleserver integriert ist. Wenn Sie keinen Wert angeben, Amazon GameLift Servers wird der Standardwert verwendet `4.0.2`. Wenn Sie eine falsche Server-SDK-Version angeben, schlägt der Spieleserver-Build möglicherweise fehl, wenn `InitSdk` Sie aufrufen, um eine Verbindung zum Amazon GameLift Servers Dienst herzustellen.
- **build-root**— Der Verzeichnispfad deiner Build-Dateien.
- **name**— Ein beschreibender Name für den neuen Build.
- **build-version**— Die Versionsdetails für die Build-Dateien.

- **region**— Die AWS Region, in der Sie Ihren Build erstellen möchten. Erstelle den Build in der Region, in der du Flotten einsetzen möchtest. Wenn du dein Spiel in mehreren Regionen einsetzt, erstelle in jeder Region einen Build.

Note

Sieh dir deine aktuelle Standardregion an, indem du [aws configure get region](#) Um Ihre Standardregion zu ändern, verwenden Sie den [aws configure set region **region name**](#) Befehl.

Beispiele

```
aws gamelift upload-build \  
  --operating-system AMAZON_LINUX_2023 \  
  --server-sdk-version "5.0.0" \  
  --build-root "~/mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-west-2
```

```
aws gamelift upload-build \  
  --operating-system WINDOWS_2022 \  
  --server-sdk-version "5.0.0" \  
  --build-root "C:\mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-west-2
```

Zeigt als Antwort auf Ihre Upload-Anfrage den Upload-Fortschritt an. Amazon GameLift Servers Amazon GameLift Servers Gibt bei einem erfolgreichen Upload die neue Build-Datensatz-ID zurück. Die Zeit für den Upload hängt von der Größe Ihrer Spieldateien und der Verbindungsgeschwindigkeit ab.

Erstellen Sie einen Build mit Dateien in Amazon S3

Sie können Ihre Build-Dateien in Amazon S3 speichern und Amazon GameLift Servers von dort hochladen. Wenn Sie Ihren Build erstellen, geben Sie den S3-Bucket-Speicherort an und Amazon GameLift Servers rufen die Build-Dateien direkt von Amazon S3 ab.

Um eine Build-Ressource zu erstellen

1. Speichern Sie Ihre Build-Dateien in Amazon S3. Erstellen Sie eine .zip-Datei mit den gepackten Build-Dateien und laden Sie sie in einen S3-Bucket in Ihrem AWS-Konto hoch. Notieren Sie sich das Bucket-Label und den Dateinamen. Sie benötigen diese, wenn Sie einen Amazon GameLift Servers Build erstellen.
2. Gewähren Sie Amazon GameLift Servers Zugriff auf Ihre Build-Dateien. Erstellen Sie eine IAM-Rolle, indem Sie den Anweisungen unter folgen. [Greifen Sie in Amazon S3 auf eine Spiele-Build-Datei zu](#) Nachdem Sie die Rolle erstellt haben, notieren Sie sich den Amazon-Ressourcennamen (ARN) der neuen Rolle. Sie benötigen ihn, wenn Sie einen Build erstellen.
3. Erstellen Sie einen Build. Verwenden Sie die Amazon GameLift Servers Konsole oder die AWS CLI , um einen neuen Build-Datensatz zu erstellen. Sie benötigen die PassRole entsprechende Genehmigung, wie unter beschrieben [Beispiele für IAM-Berechtigungen für Amazon GameLift Servers](#).

Console

1. Wählen Sie in der [Amazon GameLift ServersKonsole](#) im Navigationsbereich Hosting, Builds aus.
2. Wählen Sie auf der Seite Builds die Option Create Build aus.
3. Gehen Sie auf der Seite „Build erstellen“ unter Build-Einstellungen wie folgt vor:
 - a. Geben Sie unter Name einen Skriptnamen ein.
 - b. Geben Sie für Version eine Version ein. Da Sie den Inhalt eines Builds aktualisieren können, können Sie anhand von Versionsdaten Aktualisierungen nachverfolgen.
 - c. Wählen Sie unter Betriebssystem (OS) das Betriebssystem Ihres Gameserver-Builds aus. Du kannst diesen Wert später nicht aktualisieren.
 - d. Geben Sie für Game Server Build die S3-URI des Build-Objekts ein, das Sie auf Amazon S3 hochgeladen haben, und wählen Sie die Objektversion aus. Wenn Sie sich nicht an den Amazon S3 S3-URI und die Objektversion erinnern, wählen Sie Browse S3 und suchen Sie nach dem Build-Objekt.
 - e. Wählen Sie für die IAM-Rolle die Rolle aus, die Sie erstellt haben und die Amazon GameLift Servers Zugriff auf Ihren S3-Bucket und Ihr Build-Objekt gewährt.
4. (Optional) Fügen Sie dem Build unter Tags Tags hinzu, indem Sie Schlüssel - und Wertepaare eingeben.

5. Wählen Sie Erstellen aus.

Amazon GameLift Servers weist dem neuen Build eine ID zu und lädt die angegebene ZIP-Datei hoch. Sie können den neuen Build, einschließlich des Status, auf der Build-Seite einsehen.

AWS CLI

Verwenden Sie den [create-build](#) Befehl, um den neuen Build zu definieren und Ihre Server-Build-Dateien hochzuladen.

1. Öffnen Sie ein Befehlszeilenfenster und wechseln Sie zu einem Verzeichnis, in dem Sie den verwenden können AWS CLI.
2. Geben Sie den folgenden create-build Befehl ein:

```
aws gamelift create-build \  
  --name user-defined name of build \  
  --server-sdk-version server SDK for Amazon GameLift Servers version \  
  --operating-system supported OS \  
  --build-version user-defined build number \  
  --storage-location "Bucket"=S3 bucket label, "Key"=Build .zip file  
name, "RoleArn"=Access role ARN} \  
  --region region name
```

- name— Ein beschreibender Name für den neuen Build.
- server-sdk-version— Die Version des Server-SDK, mit Amazon GameLift Servers der Amazon GameLift Servers du deinen Spielservers integriert hast. Wenn Sie keinen Wert angeben, Amazon GameLift Servers wird der Standardwert verwendet 4.0.2.
- operating-system— Die Laufzeitumgebung des Gameserver-Builds. Sie müssen einen Betriebssystemwert angeben. Sie können dies später nicht aktualisieren.
- build-version— Die Versionsdetails für die Build-Dateien. Diese Informationen können nützlich sein, da für jede neue Version deines Spielservers eine neue Build-Ressource erforderlich ist.
- storage-location
 - Bucket— Der Name des S3-Buckets, der deinen Build enthält. Zum Beispiel „my_build_files“.
 - Key— Der Name der ZIP-Datei, die Ihre Build-Dateien enthält. Zum Beispiel „my_game_build_7.0.1, 7.0.2“.

- **RoleARN**— Der ARN, der der von Ihnen erstellten IAM-Rolle zugewiesen wurde. Zum Beispiel „arn:aws:iam: :111122223333:role/“. GameLiftAccess Eine Beispielrichtlinie finden Sie unter [Greifen Sie in Amazon S3 auf eine Spiele-Build-Datei zu](#).
- **region**— Erstellen Sie den Build in der Region, in der Sie Flotten einsetzen möchten. AWS Wenn du dein Spiel in mehreren Regionen einsetzt, erstelle in jeder Region einen Build.

 Note

Wir empfehlen, deine aktuelle Standardregion mithilfe des `configure get` Befehls zu überprüfen. Verwenden Sie den `configure set` Befehl, um Ihre Standardregion zu ändern.

Beispiel

```
aws gamelift create-build \
    --operating-system WINDOWS_2022 \
    --storage-location
    "Bucket"="my_game_build_files", "Key"="mygame_build_101.zip", "RoleArn"="arn:aws:iam::111
gamelift" \
    --name "My Game Nightly Build" \
    --build-version "build 101" \
    --region us-west-2
```

3. Verwenden Sie den `describe-build` Befehl, um den neuen Build anzuzeigen.

Erstellen Sie ein Container-Image für Amazon GameLift Servers

In diesem Thema wird beschrieben, wie Sie mit Ihrer Gameserver-Software ein Container-Image erstellen, mit dem Amazon GameLift Servers Sie es verwenden können. Ein Gameserver-Container-Image enthält die ausführbare Datei des Spieleservers und alle Abhängigkeiten, die für die Ausführung erforderlich sind. Container-Images für Spieleserver werden mit einer Hosting-Lösung für Amazon GameLift Servers verwaltete Container verwendet. Einzelheiten zum Aufbau der Komplettlösung finden Sie unter:

- [Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwalteten Containern](#)
- [So funktionieren Container in Amazon GameLift Servers](#)

Schließen Sie die folgenden Aufgaben ab, um dein Spielserver-Container-Image für den Einsatz in einer Amazon GameLift Servers Containerflotte vorzubereiten. Bevor Sie mit diesen Aufgaben beginnen, schließen Sie die Integration Ihres Spielservercodes in das Amazon GameLift Servers Server-SDK ab.

Themen

- [Erstellen Sie ein Gameserver-Container-Image](#)
- [Ein Container-Image an Amazon ECR übertragen](#)

Erstellen Sie ein Gameserver-Container-Image

Folgen Sie diesen Anweisungen, wenn Sie auf einer Linux-basierten Plattform arbeiten oder das Windows-Subsystem für Linux (WSL) mit installiertem Docker verwenden.

Um ein Gameserver-Container-Image zu erstellen

1. Bereite ein funktionierendes Verzeichnis mit deiner Gameserver-Software vor. Erstelle auf einem lokalen Computer ein Arbeitsverzeichnis, um die Dateien für deinen Gameserver-Container zu organisieren. Ihr Container-Image verwendet diese Dateistruktur, wenn es den Container für Amazon GameLift Servers Hosting-Ressourcen bereitstellt. Beispiel:

```
[~/]$ mkdir -p work/glc/gamebuild && cd work && find .  
.  
./glc  
./glc/gamebuild
```

Note

Wenn du diese Funktion ausprobierst und noch keinen funktionierenden Gameserver-Build hast, probiere unseren Beispiel-Spieleserver aus [SimpleServer](#), der auf verfügbar ist GitHub.

2. Erstelle ein neues Dockerfile mit der bereitgestellten Vorlage.
3. Folgen Sie den Anweisungen in der Dockerfile-Vorlage, um sie für Ihren eigenen Gebrauch zu aktualisieren.
 - Aktualisieren Sie das Basis-Image nach Bedarf.
 - Lege Umgebungsvariablen für deinen Gameserver-Build fest.

4. Erstellen Sie das Container-Image. Führen Sie den aus `docker build` und geben Sie Ihren eigenen Repository-Namen an. Beispiel:

```
[~/work/glc]$ docker build -t <local repository name>:<optional tag> .
```

Sie können Ihre Repositories und Bilder IDs mit dem `docker images` folgenden Befehl anzeigen, wie in diesem Beispiel dargestellt:

Dockerfile-Vorlage für ein Gameserver-Container-Image

Diese Vorlage enthält die Mindestanweisungen, die ein Spieleserver-Container benötigt, um in einer Amazon GameLift Servers Flotte verwendet zu werden. Ändern Sie den Inhalt nach Bedarf für Ihren Spieleserver.

```
# Base image
# -----
# Add the base image that you want to use,
# Make sure to use an image with the same architecture as the
# Instance type you are planning to use on your fleets.
FROM public.ecr.aws/amazonlinux/amazonlinux
#
# Game build directory
# -----
# Add your gamebuild directory to the env variable below.
# The game build provided here needs to be integrated with server sdk for Amazon
GameLift Servers.
ENV GAME_BUILD_DIRECTORY="<ADD_GAME_BUILD_DIRECTORY>" \
#
# Game executable and launch parameters
# -----
# Add the relative path to your executable in the 'GAME_EXECUTABLE' env variable
below.
# The game build provided over here needs to be integrated with server sdk for Amazon
GameLift Servers.
# This template assumes that the executable path is relative to the game build
directory.
# Add any launch parameters to pass into your executable in the 'LAUNCH_PARAMS' env
variable below.
# Add 'HOME_DIR' to identify where the game executable and logs exist.
GAME_EXECUTABLE="<ADD NAME OF EXECUTABLE WITHIN THE GAME DIRECTORY>" \
LAUNCH_PARAMS=<ADD LAUNCH PARAMETERS> \
```



```
HOME_DIR="/local/game" \  
  
# Install dependencies as necessary  
RUN yum install -y shadow-utils  
  
RUN mkdir -p $HOME_DIR  
COPY ./$GAME_BUILD_DIRECTORY/ $HOME_DIR  
  
# Change directory to home  
WORKDIR $HOME_DIR  
  
# Set up for 'gamelift' user  
RUN useradd -m gamelift && \  
    echo "gamelift ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers && \  
    chown -R gamelift:gamelift $HOME_DIR  
  
# Add permissions to game build  
RUN chmod +x ./$GAME_EXECUTABLE  
  
USER gamelift  
  
# Check directory before starting the container  
RUN ls -lhrt .  
  
# Check path before starting the container  
RUN echo $PATH  
  
# Start the game build  
ENTRYPOINT ["/bin/sh", "-c", "./$GAME_EXECUTABLE", "$LAUNCH_PARAMS"]
```

Ein Container-Image an Amazon ECR übertragen

Nachdem Sie ein Container-Image für die Bereitstellung erstellt haben, speichern Sie das Image in einem öffentlichen oder privaten Repository in Amazon ECR. Diesem Repository wird ein URI-Wert zugewiesen, mit dem Amazon GameLift Servers ein Snapshot des Images für die Bereitstellung in einer Containerflotte erstellt wird.

Note

Wenn Sie noch kein privates Amazon ECR-Repository haben, [erstellen Sie eines](#).

Um Ihr Container-Image an Amazon ECR zu übertragen

1. Holen Sie sich Ihre Amazon ECR-Anmeldeinformationen. Bevor Sie ein Container-Image an Amazon ECR übertragen, müssen Sie zunächst Ihre AWS Anmeldeinformationen in temporärer Form abrufen und sie Docker zur Verfügung stellen. Docker benötigt diese Anmeldeinformationen, um sich anzumelden.

```
[~/work/glc]$ aws ecr get-login-password --region us-west-2 | docker login --  
username AWS --password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com  
WARNING! Your password will be stored unencrypted in  
/home/user-name/.docker/config.json.  
Configure a credential helper to remove this warning.  
See https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
  
Login Succeeded
```

2. Kopieren Sie die URI des [privaten Amazon ECR-Repositorys](#), das Sie verwenden möchten.
3. Wenden Sie ein Amazon ECR-Tag auf Ihr Container-Image an.

Example

```
[~/work/glc]$ docker tag <IMAGE ID from above> <Amazon ECR private repository  
URI>:<optional tag>
```

4. Übertragen Sie Ihr Container-Image auf Amazon ECR

Example

```
[~/work/glc]$ docker image push <Amazon ECR private repository URI>
```

client/server Spielinteraktionen mit Amazon GameLift Servers

Die Komponenten in Ihrer Amazon GameLift Servers Hosting-Lösung interagieren auf spezifische Weise miteinander, um Spielsitzungen als Reaktion auf die Nachfrage der Spieler durchzuführen. In diesem Thema wird beschrieben, wie Komponenten miteinander kommunizieren, wenn Ihr Spieleserver auf Amazon GameLift Servers verwalteten EC2-Flotten, selbstverwalteten Amazon GameLift Servers Anywhere-Flotten oder einer Hybridlösung gehostet wird.

Zu den Komponenten der Hosting-Lösung gehören ein Spieleserver, der Amazon GameLift Servers Dienst, ein clientseitiger Backend-Dienst und ein Spieleclient. Der Spieleserver verwendet das

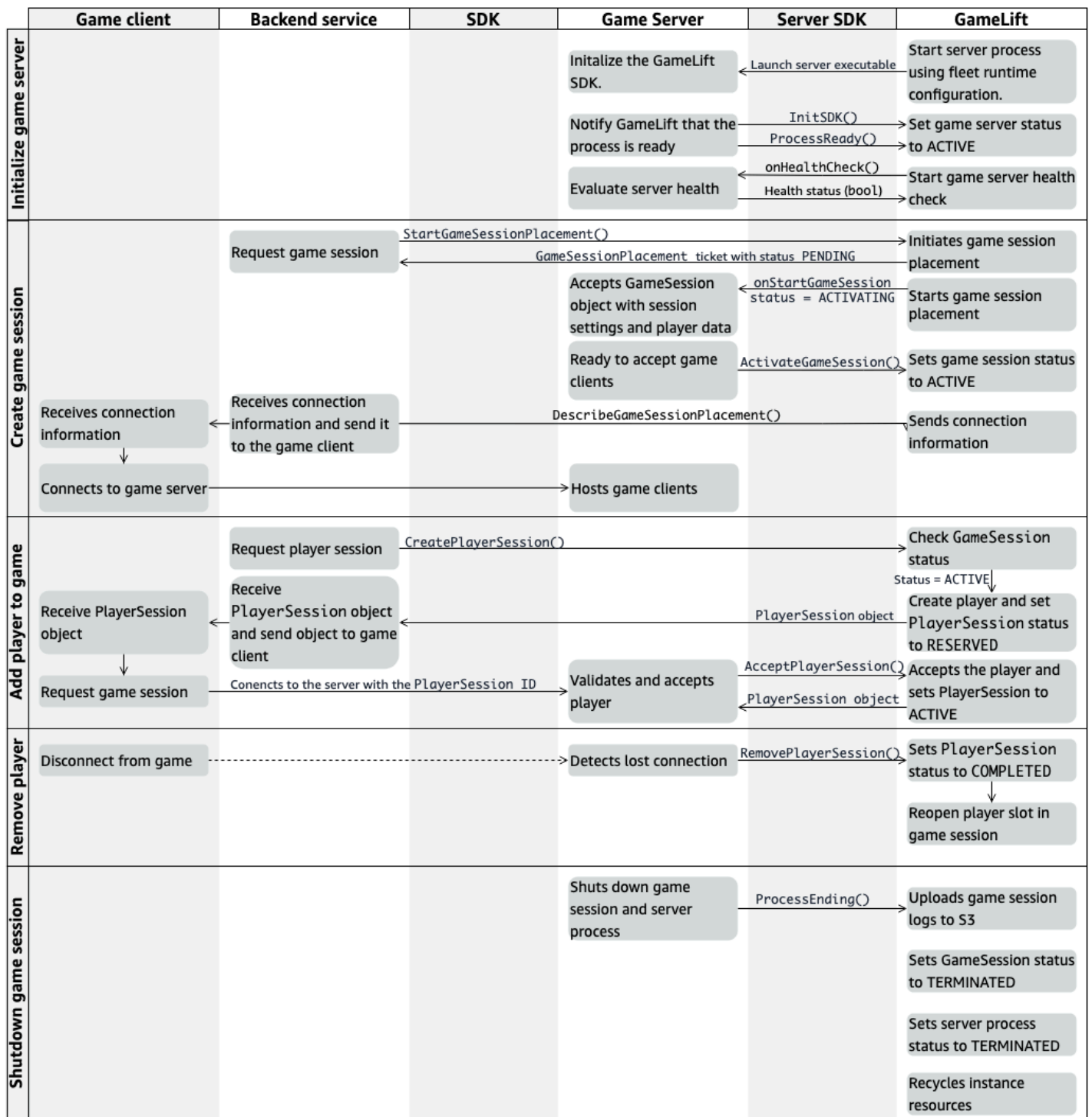
[Amazon GameLift ServersServer-SDK](#), um mit dem Dienst zu interagieren. Amazon GameLift Servers Der Backend-Dienst verwendet die [Amazon GameLift ServersService-API](#) (Teil des AWS SDK), um im Namen des Spielclients mit dem Dienst zu interagieren. Wenn Sie einer Spielsitzung beitreten, stellt der Spielclient über die eindeutige IP-Adresse und Portnummer der Spielsitzung eine direkte Verbindung zu einer Spielsitzung her.

Themen

- [Diagramm der Interaktionen](#)
- [Verhalten bei Interaktionen](#)

Diagramm der Interaktionen

Das folgende Diagramm zeigt, wie Ihre Spielhosting-Komponenten interagieren, sodass der Amazon GameLift Servers Dienst den Status der Verfügbarkeit von Spieleservern verfolgen und Spielsitzungen als Reaktion auf Spieleranforderungen starten kann.



Verhalten bei Interaktionen

In den folgenden Abschnitten wird die Reihenfolge der Ereignisse in jeder der wichtigsten Interaktionen beschrieben.

Initialisierung eines Gameserver-Prozesses

Beim Start stellt ein Spieleserverprozess die Kommunikation mit dem Amazon GameLift Servers Dienst her und meldet seinen Status als bereit, eine Spielsitzung zu veranstalten.

1. Ein neuer Prozess der ausführbaren Datei des Spieleservers wird auf einer Hosting-Ressource ausgeführt.
2. Der Gameserver-Prozess ruft nacheinander die folgenden Server-SDK-Operationen auf:
 - a. `InitSDK()` um das Server-SDK zu initialisieren, den Serverprozess zu authentifizieren und die Kommunikation mit dem Amazon GameLift Servers Dienst herzustellen.
 - b. `ProcessReady()` um die Bereitschaft zu kommunizieren, eine Spielsitzung zu veranstalten. Dieser Aufruf meldet auch die Verbindungsinformationen des Prozesses, die Spielclients verwenden, um sich mit der Spielsitzung zu verbinden, und andere Informationen.

Der Serverprozess wartet dann auf Eingabeaufforderungen des Amazon GameLift Servers Dienstes.

3. Amazon GameLift Servers aktualisiert den Status des Serverprozesses auf ACTIVE und ist für das Hosten einer neuen Spielsitzung verfügbar.
4. Amazon GameLift Servers beginnt in regelmäßigen Abständen, den `onHealthCheck` Callback aufzurufen, um einen Integritätsstatus von den Serverprozessen abzufragen. Diese Aufrufe werden fortgesetzt, solange der Serverprozess im aktiven Status bleibt. Der Serverprozess muss innerhalb einer Minute fehlerfrei oder fehlerfrei antworten. Wenn der Serverprozess nicht ordnungsgemäß reagiert oder nicht reagiert, ändert der Amazon GameLift Servers Dienst irgendwann den aktiven Status des Serverprozesses und sendet keine Anfragen mehr, um die Spielsitzung zu starten.

Eine Spielsitzung erstellen

Der Amazon GameLift Servers Dienst startet eine neue Spielsitzung als Antwort auf eine Anfrage eines Spielers, das Spiel zu spielen.

1. Ein Spieler, der den Spielclient verwendet, bittet darum, an einer Spielsitzung teilzunehmen. Je nachdem, wie dein Spiel den Spielerbeitritt handhabt, sendet der Spielclient eine Anfrage an den Backend-Dienst.
2. Wenn für den Spielerbeitritt eine neue Spielsitzung gestartet werden muss, sendet der Backend-Dienst eine Anfrage für eine neue Spielsitzung an den Amazon GameLift

Servers Dienst. Diese Anfrage ruft den API-Vorgang `StartGameSessionPlacement()` des Dienstes auf. (Als Alternative könnte der Back-End-Dienst oder aufrufen `StartMatchmaking()CreateGameSession()`.)

3. Der Amazon GameLift Servers Dienst reagiert darauf, indem er ein neues `GameSessionPlacement` Ticket mit Status `PENDING` erstellt. Er sendet Ticketinformationen an den Backend-Service zurück, sodass dieser den Status des Platzierungstickets verfolgen und feststellen kann, wann die Spielsitzung für Spieler bereit ist. Weitere Informationen finden Sie unter [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#).
4. Der Amazon GameLift Servers Dienst startet den Platzierungsprozess für die Spielsitzung. Er identifiziert, welche Flotten untersucht werden müssen, und durchsucht diese Flotten nach einem aktiven Serverprozess, der keine Spielsitzung hostet. Bei der Suche nach einem verfügbaren Serverprozess geht der Amazon GameLift Servers Dienst wie folgt vor:
 - a. Erstellt ein `GameSession` Objekt mit den Einstellungen für die Spielsitzung und den Spielerdaten aus der Platzierungsanfrage und setzt den Status auf `ACTIVATING`.
 - b. Fordert den Serverprozess auf, eine Spielsitzung zu starten. Der Dienst ruft den `onStartGameSession` Callback des Serverprozesses auf und übergibt das Objekt `GameSession`.
 - c. Ändert die Anzahl der Spielsitzungen des Serverprozesses auf 1.
5. Der Serverprozess führt seine `onStartGameSession` Callback-Funktion aus. Wenn der Serverprozess bereit ist, Player-Verbindungen anzunehmen, ruft er den Server-SDK-Vorgang auf `ActivateGameSession()` und wartet auf Player-Verbindungen.
6. Der Amazon GameLift Servers Dienst aktualisiert das `GameSession` Objekt mit Verbindungsinformationen für den Serverprozess (wie im Call to `angegebenProcessReady()`) und setzt den Status der Spielsitzung auf `ACTIVE`. Außerdem aktualisiert er den `GameSessionPlacement` Ticketstatus auf `FULFILLED`.
7. Der Backend-Dienst ruft `DescribeGameSessionPlacement()` an, um den Ticketstatus zu überprüfen und Informationen zur Spielsitzung abzurufen. Wenn die Spielsitzung aktiv ist, benachrichtigt der Backend-Dienst den Spielclient und leitet die Verbindungsinformationen der Spielsitzung weiter.
8. Der Spielclient verwendet die Verbindungsinformationen, um sich direkt mit dem Spielserver-Prozess zu verbinden und an der Spielsitzung teilzunehmen.

Einen Spieler zu einem Spiel hinzufügen

Ein Spiel kann optional Spilersitzungen verwenden, um Spielerverbindungen zu Spielsitzungen nachzuverfolgen. Spilersitzungen können einzeln oder als Teil einer Platzierungsanfrage für Spielsitzungen erstellt werden.

1. Der Backend-Dienst ruft den API-Vorgang des Dienstes `CreatePlayerSession()` mit einer Spielsitzungs-ID auf.
2. Der Amazon GameLift Servers Dienst überprüft den Status der Spielsitzung (muss vorhanden sein `ACTIVE`) und sucht in der Spielsitzung nach einem offenen Spielerplatz. Wenn ein Spielautomat verfügbar ist, geht der Dienst wie folgt vor:
 - a. Erstellt ein neues `PlayerSession` Objekt und setzt den Status auf `RESERVED`.
 - b. Reagiert auf die Backend-Serviceanfrage mit Informationen zur Spilersitzung.
3. Der Backend-Dienst leitet die Sitzungsinformationen des Spielers zusammen mit den Verbindungsinformationen der Spielsitzung an den Spielclient weiter.
4. Der Spielclient verwendet die Verbindungsinformationen und die Sitzungs-ID des Spielers, um eine direkte Verbindung zum Spielserverprozess herzustellen und um die Teilnahme an der Spielsitzung zu bitten.
5. Als Reaktion auf einen Beitrittsversuch des Spielclients ruft der Spieleserver-Prozess den API-Vorgang des Dienstes auf, `AcceptPlayerSession()` um die Sitzungs-ID des Spielers zu überprüfen. Der Serverprozess akzeptiert die Verbindung oder weist sie zurück.
6. Der Amazon GameLift Servers Dienst führt eine der folgenden Aktionen aus:
 - a. Wenn die Verbindung akzeptiert wird, Amazon GameLift Servers setzt er den `PlayerSession` Status auf `ACTIVE` und leitet den Prozess `PlayerSession` an den Spieleserver weiter.
 - b. Wenn der Spieleserver-Prozess die Sitzungs-ID des Spielers nicht innerhalb eines bestimmten Zeitraums nach der ursprünglichen `CreatePlayerSession()` Anfrage abfragt, ändert der Amazon GameLift Servers Dienst den `PlayerSession` Status in `TIMEDOUT` und öffnet den Spieler-Slot in der Spielsitzung erneut. `AcceptPlayerSession()`

Einen Spieler entfernen

Bei Spielen, die Spilersitzungen verwenden, benachrichtigt der Spieleserverprozess den Amazon GameLift Servers Dienst, wenn ein Spieler die Verbindung trennt. Der Dienst verwendet diese

Informationen, um den Status von Spieler-Slots in einer Spielsitzung zu verfolgen und kann neuen Spielern die Nutzung offener Spielautomaten ermöglichen.

1. Ein Spieler trennt die Verbindung zur Spielsitzung.
2. Der Spieleserver-Prozess erkennt die unterbrochene Verbindung und ruft den Server-SDK-Vorgang `RemovePlayerSession()` auf.
3. Der Amazon GameLift Servers Dienst ändert den Status der Spielsitzung auf `COMPLETED` und öffnet den Spieler-Slot in der Spielsitzung erneut.

Die Spielsitzung wird beendet

Am Ende einer Spielsitzung oder beim Beenden der Spielsitzung informiert der Serverprozess den Amazon GameLift Servers Dienst über den Status der Spielsitzung.

1. Der Spieleserverprozess beendet die Spielsitzung und leitet das Herunterfahren des Prozesses ein, indem er den Server-SDK-Vorgang aufruft. `ProcessEnding()`
2. Der Spieleserver-Prozess ruft `Destroy()` auf, um das Server-SDK aus dem Speicher zu befreien. Dieser Schritt ist für [Telemetriemetriemetriken](#) erforderlich, um zu verhindern, dass normale Prozessabbrüche als Abstürze gemeldet werden.
3. Der Amazon GameLift Servers Dienst macht Folgendes:
 - a. Lädt Spielsitzungsprotokolle auf Amazon Simple Storage Service (Amazon S3) hoch.
 - b. Ändert den Status der Spielsitzung auf `TERMINATED`.
 - c. Ändert den Status des Serverprozesses auf `TERMINATED`.
 - d. Je nachdem, wie die Hosting-Lösung konzipiert ist, werden die neu verfügbaren Hosting-Ressourcen für die Ausführung eines neuen Spieleserver-Prozesses zugewiesen.

Erstellen Sie einen Backend-Service für Amazon GameLift Servers

Wir empfehlen dir, einen Spiele-Client-Dienst zu implementieren, der deine Spieler authentifiziert und mit der API kommuniziert. Amazon GameLift Servers Durch die Implementierung eines benutzerdefinierten Spiele-Client-Dienstes kannst du:

- Passen Sie die Authentifizierung für Ihre Spieler an.
- Steuere, wie Spieler für neue Spielsitzungen Amazon GameLift Servers gruppiert oder zu bestehenden Spielsitzungen hinzugefügt werden sollen.

- Sammle Informationen aus deinen eigenen Ressourcen, um Sesplayer-Attribute für das Spiel bereitzustellen, wie z. B. die Fähigkeitsbewertung, für die Spielersuche, anstatt dem Client zu vertrauen.

Die Verwendung eines Spiele-Client-Dienstes reduziert auch die Sicherheitsrisiken, die durch direkte Interaktionen von Spieleclients mit deiner API entstehen. Amazon GameLift Servers

Authentifizieren Sie Ihre Spieler

Sie können Amazon Cognito und Player Session verwenden, IDs um Ihre Spieleclients zu authentifizieren. Verwenden Sie Amazon Cognito Cognito-Benutzerpools, um den Lebenszyklus und die Eigenschaften Ihrer Spieleridentitäten zu verwalten.

Wenn Sie es vorziehen, können Sie eine benutzerdefinierte Identitätslösung erstellen und diese dort hosten. AWS Sie können Lambda-Autorisierer auch für eine benutzerdefinierte Autorisierungslogik mit API Gateway verwenden.

Weitere Ressourcen:

- [Verwenden von Identitätspools \(förderierte Identitäten\)](#) (Amazon Cognito Developer Guide)
- [Erste Schritte mit Benutzerpools](#) (Amazon Cognito Developer Guide)
- [So richten Sie die Spielerauthentifizierung mit Amazon Cognito ein](#) (AWS für Spieleblog)

Integrieren Sie die Amazon GameLift Servers Game-Client-Funktionalität

Integrieren Sie Funktionen zum Hosten von Amazon GameLift Servers Spielen in alle Lösungskomponenten, die mit dem Amazon GameLift Servers Dienst kommunizieren müssen, z. B. um Informationen zu Spielsitzungen abzurufen oder neue Spielsitzungen zu erstellen. In den meisten Fällen ist diese Funktionalität in die Backend-Servicekomponente integriert. Fügen Sie Funktionen für Kernaufgaben hinzu, darunter:

- Fordere Informationen und den Status aktiver Spielsitzungen an.
- Reservieren Sie einen Platz für einen neuen Spieler in einer bestehenden Spielsitzung.
- Erstelle eine neue Spielsitzung für eine Gruppe von Spielern.
- Anfordern von Matchmaking für einen oder mehrere Spieler.
- Stellen Sie aktualisierte Daten für bestehende Spielsitzungen bereit.

Weitere Informationen [client/server Spielinteraktionen mit Amazon GameLift Servers](#) zur Interaktion der Komponenten zum Hosten von Spielen finden Sie unter Amazon GameLift Servers SDKs.

Richten Sie die Amazon GameLift Servers API ein

Fügen Sie das AWS SDK für C++ mit Amazon GameLift Servers einem Projekt hinzu.

Fügen Sie Code hinzu, um einen Amazon GameLift Servers Client zu initialisieren und wichtige Einstellungen zu speichern. Dieser Code muss vor jedem Code ausgeführt werden, der Amazon GameLift Servers von abhängt.

1. Richten Sie eine Client-Konfiguration ein. Verwenden Sie die Standard-Client-Konfiguration oder erstellen Sie ein benutzerdefiniertes Client-Konfigurationsobjekt. Weitere Informationen finden Sie unter [AWS::Client::ClientConfiguration](#)(C++) oder [AmazonGameLiftConfig](#)(C#).

Eine Client-Konfiguration gibt eine Zielregion und einen Endpunkt an, die bei der Kontaktaufnahme Amazon GameLift Servers verwendet werden sollen. Die Region gibt die Menge der bereitgestellten Ressourcen (Flotten, Warteschlangen und Matchmaker) an, die verwendet werden sollen. In der Standard-Client-Konfiguration ist der Standort auf die Region USA Ost (Nord-Virginia) festgelegt. Um eine andere Region zu verwenden, erstellen Sie eine benutzerdefinierte Konfiguration.

2. Initialisieren eines Amazon GameLift Servers-Clients. Verwenden Sie [AwsGameLift::GameLiftClient\(\)](#) (C++) oder [AmazonGameLiftClient\(\)](#) (C#) mit einer Standard-Client-Konfiguration oder einer benutzerdefinierten Client-Konfiguration.
3. Wenn Sie Spielsitzungen verwenden, fügen Sie einen Mechanismus hinzu, um für jeden Spieler eine eindeutige Kennung zu generieren. Weitere Informationen finden Sie unter [Spieler generieren IDs](#).
4. Erfassen und speichern Sie die folgenden Informationen:
 - Zielflotte — Wenn du Spielsitzungen für eine bestimmte Flotte manuell erstellst, gib eine Flotten-ID oder eine Alias-ID an, die auf die Zielflotte verweist. Wenn es sich bei der Flotte um eine Flotte mit mehreren Standorten handelt, geben Sie einen Flottenstandort an. Es hat sich bewährt, Flottenaliasse zu verwenden, damit Sie Spieler von einer Flotte zur anderen wechseln können, ohne Ihren Backend-Service aktualisieren zu müssen.
 - Zielwarteschlange — Geben Sie für Spiele, die Warteschlangen mit mehreren Flotten verwenden, um neue Spielsitzungen zu platzieren, den Namen der Warteschlange an, an die Platzierungsanfragen gesendet werden sollen.

- **AWS Anmeldedaten** — Bei allen Anrufen Amazon GameLift Servers müssen die Zugangsdaten der Person angegeben werden AWS-Konto, die das Spiel hostet. Sie können diese Anmeldeinformationen erwerben, indem Sie einen Spielerbenutzer erstellen, wie unter beschrieben [Richte den programmatischen Zugriff für dein Spiel ein](#). Gehen Sie je nachdem, wie Sie den Zugriff für den Player-Benutzer verwalten, wie folgt vor:
 - Wenn Sie eine Rolle zur Verwaltung von Spielerbenutzerberechtigungen verwenden, fügen Sie Code hinzu, um die Rolle zu übernehmen, bevor Sie eine Amazon GameLift Servers API aufrufen. Die Anfrage zur Übernahme der Rolle gibt einen Satz temporärer Sicherheitsanmeldedaten zurück. Weitere Informationen finden Sie unter [Wechseln zu einer IAM-Rolle \(AWS API\)](#) im IAM-Benutzerhandbuch.
 - Wenn Sie über langfristige Sicherheitsanmeldedaten verfügen, konfigurieren Sie Ihren Code so, dass gespeicherte Anmeldeinformationen gefunden und verwendet werden. Weitere Informationen finden Sie unter [Authentifizieren mit langfristigen Anmeldeinformationen](#) im AWS SDKs Referenzhandbuch zu Tools. Informationen zum Speichern von Anmeldeinformationen finden Sie in den AWS API-Referenzen für [\(C++\)](#) und [\(.NET\)](#).
 - Wenn Sie über temporäre Sicherheitsanmeldedaten verfügen, fügen Sie mithilfe von AWS -Security-Token-Service (AWS STS) Code hinzu, um die Anmeldeinformationen regelmäßig zu aktualisieren, wie im IAM-Benutzerhandbuch [unter Verwenden temporärer Sicherheitsanmeldedaten mit dem AWS SDKs](#) beschrieben. Der Code muss neue Anmeldeinformationen anfordern, bevor die alten ablaufen.

Holen Sie sich aktive Spielsitzungen

Füge Code hinzu, um verfügbare Spielsitzungen zu entdecken und die Einstellungen und Metadaten der Spielsitzungen zu verwalten.

Suche nach aktiven Spielsitzungen

[SearchGameSessions](#) Dient zum Abrufen von Informationen zu einer bestimmten Spielsitzung, zu allen aktiven Sitzungen oder zu Sitzungen, die einer Reihe von Suchkriterien entsprechen. Dieser Aufruf gibt für jede aktive Spielsitzung ein [GameSession](#) Objekt zurück, das Ihrer Suchanfrage entspricht. Dieses Objekt enthält den DNS-Namen, die IP-Adresse und den Port, die ein Spielclient benötigt, um eine Verbindung zur Spielsitzung herzustellen.

Verwenden Sie Suchkriterien, um eine gefilterte Liste aktiver Spielsitzungen abzurufen, an denen Spieler teilnehmen können. Beispielsweise können Sie Sitzungen wie folgt filtern:

- Schließt volle Spielsitzungen aus: `CurrentPlayerSessionCount = MaximumPlayerSessionCount`.
- Wählen Sie Spielsitzungen basierend auf der Dauer der Sitzung aus: `AuswertenCreationTime`.
- Finden Sie Spielsitzungen, die auf einer benutzerdefinierten Spieleigenschaft basieren: `gameSessionProperties.gameMode = "brawl"`.

Spielsitzungsdaten verwalten

Verwenden Sie eine der folgenden Operationen, um Informationen zu Spielsitzungen abzurufen oder zu aktualisieren.

- [DescribeGameSessionDetails\(\)](#) — Ruft zusätzlich zu den Informationen zur Spielsitzung den Schutzstatus einer Spielsitzung ab.
- [UpdateGameSession\(\)](#) — Ändern Sie die Metadaten und Einstellungen einer Spielsitzung nach Bedarf.
- [GetGameSessionLogUrl](#) — Greifen Sie auf gespeicherte Spielsitzungsprotokolle zu.

Spielsitzungen erstellen

Fügen Sie Code zum Starten neuer Spielsitzungen auf den von Ihnen bereitgestellten Flotten hinzu und stellen Sie diese Spielern zur Verfügung. Es gibt zwei Möglichkeiten, Spielsitzungen zu erstellen, je nachdem, wie Ihre Game-Hosting-Lösung die Platzierungen von Spielsitzungen verwaltet.

Erstellen Sie eine Spielsitzung mit einer Warteschlange mit mehreren Standorten

Wird verwendet [StartGameSessionPlacement](#), um eine Anfrage für eine neue Spielsitzung in eine Warteschlange zu stellen. Um diesen Vorgang zu verwenden, erstellen Sie eine Warteschlange. Dies bestimmt, Amazon GameLift Servers wo die neue Spielsitzung stattfindet. Weitere Informationen zu Warteschlangen und deren Verwendung finden Sie unter [Platzierung der Spielsitzungen konfigurieren](#).

Wenn Sie eine Platzierung für Spielsitzungen erstellen, geben Sie den Namen der zu verwendenden Warteschlange, einen Namen für die Spielsitzung, eine maximale Anzahl gleichzeitiger Spieler und einen optionalen Satz von Spieleigenschaften an. Sie können optional auch eine Liste von Spielern angeben, die automatisch an der Spielsitzung teilnehmen sollen. Wenn du Daten zur Spielerlatenz für relevante Regionen hinzufügst, Amazon GameLift Servers verwendet Then diese Informationen, um die neue Spielsitzung auf einer Flotte zu platzieren, die den Spielern das ideale Spielerlebnis bietet.

Verwenden Sie die UDP-Ping-Beacons, um Amazon GameLift Servers genaue Latenzmessungen zu erhalten. Mit diesen Endpunkten können Sie die tatsächliche UDP-Netzwerklatenz zwischen Player-Geräten und potenziellen Hosting-Standorten messen, was zu genaueren Platzierungsentscheidungen führt als mit ICMP-Pings. Weitere Informationen zur Verwendung von UDP-Ping-Beacons zur Messung der Latenz finden Sie unter [UDP-Ping-Beacons](#)

Die Platzierung von Spielsitzungen ist ein asynchroner Prozess. Nachdem Sie eine Anfrage gestellt haben, können Sie sie erfolgreich ausführen oder das Zeitlimit überschreiten. Sie können die Anfrage auch jederzeit stornieren, indem Sie [StopGameSessionPlacement](#). Rufen Sie an, um den Status Ihrer Platzierungsanfrage zu überprüfen [DescribeGameSessionPlacement](#).

Erstelle eine Spielsitzung auf einer bestimmten Flotte

Wird verwendet [CreateGameSession](#), um eine neue Sitzung auf einer bestimmten Flotte zu erstellen. Diese synchrone Operation gelingt oder schlägt fehl, abhängig davon, ob die Flotte über Ressourcen für das Hosten einer neuen Spielsitzung verfügt. Nachdem Sie die neue Spielsitzung Amazon GameLift Servers erstellt und ein [GameSession](#) Objekt zurückgegeben haben, können Sie ihr Spieler hinzufügen.

Wenn Sie diesen Vorgang verwenden, geben Sie eine Flotten- oder Alias-ID, einen Sitzungsnamen und die maximale Anzahl gleichzeitiger Spieler für das Spiel an. Optional können Sie eine Gruppe von Spieleigenschaften einschließen. Spieleigenschaften werden in einer Reihe von Schlüssel-Wert-Paaren definiert.

Wenn Sie die Amazon GameLift Servers Ressourcenschutzfunktion verwenden, um die Anzahl der Spielsitzungen zu begrenzen, die ein Spieler erstellen kann, geben Sie die Spieler-ID des Erstellers der Spielsitzung an.

Schließe einen Spieler an einer Spielsitzung an

Füge Code hinzu, um einen Spielerplatz in einer aktiven Spielsitzung zu reservieren und Spielclients mit Spielsitzungen zu verbinden. Diese Aktion ist verfügbar, wenn dein Spiel Spielsitzungen mit einem einzelnen Spieler verwendet IDs. Weitere Informationen zu Spielsitzungen finden Sie unter [Amazon GameLift Servers und das Spielerlebnis](#).

1. Reservieren Sie einen Spielerplatz in einer Spielsitzung

Um einen Spielerplatz zu reservieren, erstellen Sie eine Spielsitzung für die Spielsitzung. Es gibt zwei Möglichkeiten dafür:

- Wenn Sie eine Spielsitzung verwenden [StartGameSessionPlacement](#) oder erstellen [StartMatchmaking](#) möchten, können Sie einen oder mehrere Spieler in die Anfrage aufnehmen, um in der neuen Spielsitzung Spielsitzungen für sie zu erstellen.
- Um Spieler zu einer bestehenden Spielsitzung hinzuzufügen, rufen Sie an [CreatePlayerSession](#) oder [CreatePlayerSessions](#) geben Sie eine Spielsitzungs-ID an.

Eine Anfrage für eine Spielsitzung muss eine eindeutige Spieler-ID enthalten. Weitere Informationen finden Sie unter [Spieler generieren IDs](#). Bei Erhalt der Anfrage wird Amazon GameLift Servers überprüft, ob die Spielsitzung neue Spieler akzeptiert und ob Spielplätze verfügbar sind. Falls erfolgreich, Amazon GameLift Servers reserviert es einen Spielplatz für den Spieler, erstellt die neue Spielsitzung und gibt ein [PlayerSession](#) Objekt zurück.

Eine Spielsitzung kann eine Reihe von benutzerdefinierten Spielerdaten enthalten. Diese Daten werden im neu erstellten Spielsitzungsobjekt gespeichert. Amazon GameLift Servers übergibt dieses Objekt an den Spieleserver, wenn der Spieler eine direkte Verbindung zur Spielsitzung herstellt. Wenn du mehrere Spielsitzungen anforderst, ordne jede Zeichenfolge von Spielerdaten einer Spieler-ID zu.

2. Mit einer Spielsitzung verbinden

Fügen Sie dem Backend-Dienst Code hinzu, um das `PlayerSession` Objekt abzurufen (z. B. durch Aufrufen von [DescribePlayerSessions\(\)](#)) und übergeben Sie es an den Spielclient zurück. Dieses Objekt enthält den DNS-Namen, die IP-Adresse und den Port. Der Spielclient kann diese Informationen verwenden, um eine direkte Verbindung zum Server herzustellen.

- Sie können eine Verbindung über den angegebenen Port und den DNS-Namen oder die IP-Adresse herstellen, die dem Serverprozess zugewiesen wurden.
- Wenn für Ihre Flotten die Generierung von TLS-Zertifikaten aktiviert ist, stellen Sie mithilfe des DNS-Namens und -Ports eine Verbindung her.
- Wenn dein Spieleserver eingehende Spielerverbindungen validiert, gib die Sitzungs-ID des Spielers an.

Nach dem Herstellen der Verbindung kommunizieren der Spielclient und der Serverprozess direkt und ohne Beteiligung Amazon GameLift Servers. Der Server unterhält die Kommunikation mit dem Spieler, Amazon GameLift Servers um den Verbindungsstatus, den Gesundheitszustand und mehr zu melden.

Wenn der Spielserver eingehende Spieler bestätigt, überprüft er, ob die Sitzungs-ID des Spielers mit einem reservierten Slot in der Spielsitzung übereinstimmt, und akzeptiert oder verweigert die Spielerverbindung. Wenn der Spieler die Verbindung trennt, meldet der Serverprozess den Verbindungsabbruch.

Übergeben Sie benutzerdefinierte Spieldaten an eine Spielsitzung

Ihr Spielclient kann Daten an eine Spielsitzung weitergeben. Wenn Sie eine Spielsitzung erstellen, können Sie eine Reihe von Spieleigenschaften (Schlüssel-Wert-Paare) oder Spielsitzungsdaten (Zeichenkettenwerte) in die Anfrage aufnehmen. Sie können auch eine bestehende Spielsitzung mit neuen oder aktualisierten Spieldaten aktualisieren. Diese Daten werden an den Spielserver-Prozess weitergegeben, der die Spielsitzung hostet, zur Verwendung durch den Spielservercode. Spieleigenschaften können nicht gelöscht werden.

Nehmen wir als Beispiel an, dass Ihr Spiel die Schwierigkeitsgrade definiert:

NoviceEasy,Intermediate, undExpert. Ein Spieler entscheidet sich dafürEasy, einem Spiel beizutreten. Ihr Spielclient fordert über den Backend-Dienst eine neue Spielsitzung mit der folgenden Spieleigenschaft an:{"Key": "Difficulty", "Value":"Easy"}. Amazon GameLift ServersFordert daraufhin einen verfügbaren Spieleserver auf, eine neue Spielsitzung zu starten, und übergibt das GameSession Objekt. Der Spielserver-Prozess verwendet die bereitgestellte Spieleigenschaft, um den Schwierigkeitsgrad der Spielsitzung festzulegen.

Weitere Informationen

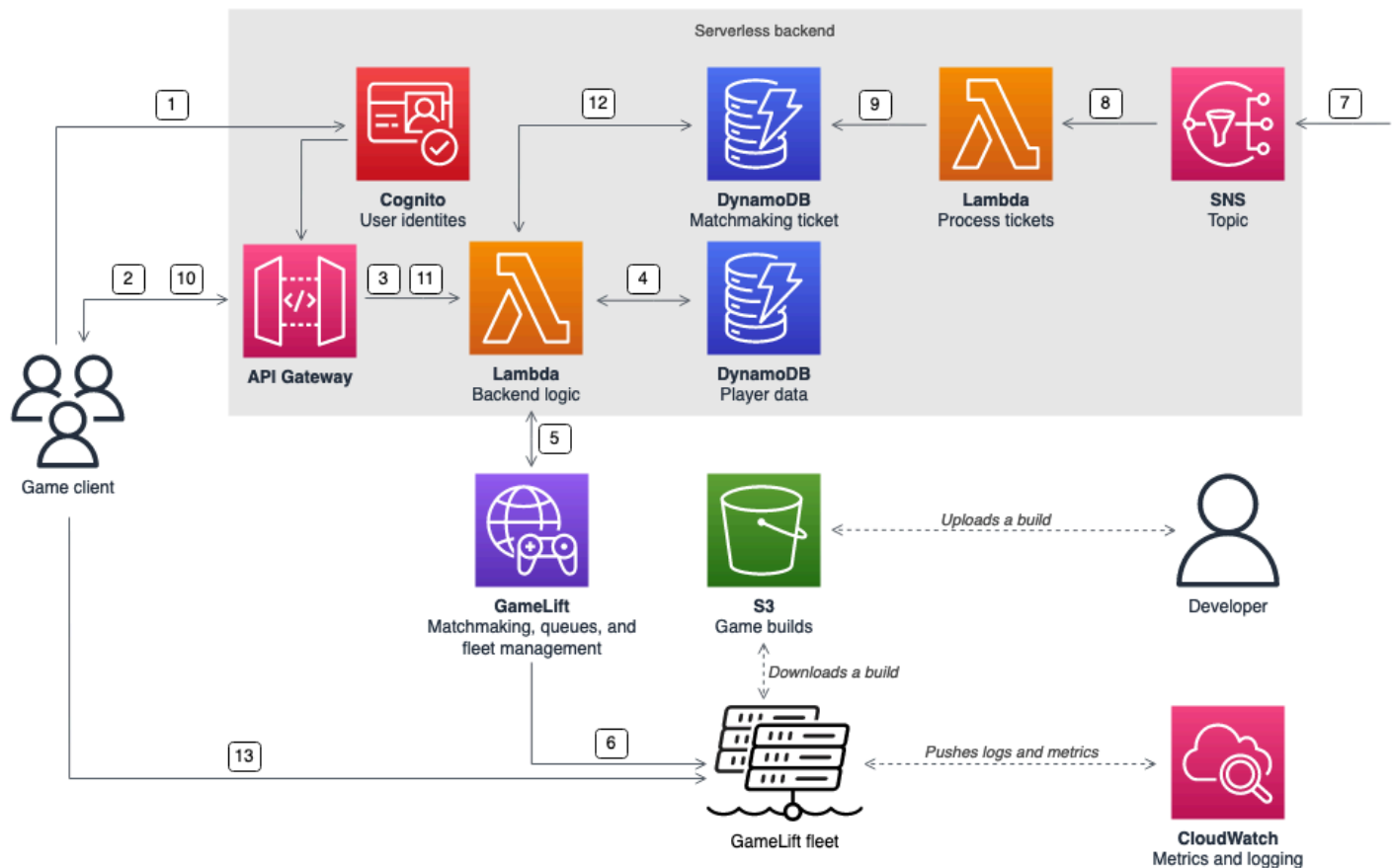
- [GameProperty Datentyp](#)
- [SearchGameSessions\(\) Beispiele](#)
- [UpdateGameSession\(\) GameProperties Parameter](#)

Eigenständige Spielesitzungsserver mit einem serverlosen Backend

Mithilfe einer serverlosen Client-service-Architektur kann das Backend den Status von Matchmaking-Tickets aus einer hoch skalierbaren Datenbank abrufen, anstatt direkt auf die API zuzugreifen.
Amazon GameLift Servers

Das folgende Diagramm zeigt ein serverloses Backend, mit dem Spieler Spiele zugeordnet werden können AWS-Services , die auf Flotten laufen. Amazon GameLift Servers Die folgende Liste enthält

eine Beschreibung für jeden nummerierten Callout im Diagramm. Um dieses Beispiel auszuprobieren, lesen Sie den Artikel [Sitzungsbasiertes Spielen für mehrere Spieler auf AWS](#) [GitHub](#)



1. Der Spielclient fordert eine Amazon Cognito Cognito-Benutzeridentität aus einem Amazon Cognito Cognito-Identitätspool an.
2. Der Spielclient erhält temporäre Zugangsdaten und fordert über eine Amazon API Gateway eine Spielsitzung an.
3. API Gateway ruft eine AWS Lambda Funktion auf.
4. Die Lambda-Funktion fordert Spielerdaten aus einer Amazon DynamoDB DynamoDB-NoSQL-Tabelle an. Die Funktion stellt die Amazon Cognito Cognito-Identität in den Anforderungskontextdaten bereit.
5. Die Lambda-Funktion fordert über Amazon GameLift Servers FlexMatch Matchmaking eine Übereinstimmung an.
6. FlexMatchordnet eine Gruppe von Spielern mit geeigneter Latenz zu und fordert dann über eine Warteschlange eine Platzierung für eine Spielsitzung anAmazon GameLift Servers. Die Warteschlange enthält Flotten mit einem oder mehreren AWS-Region Standorten.

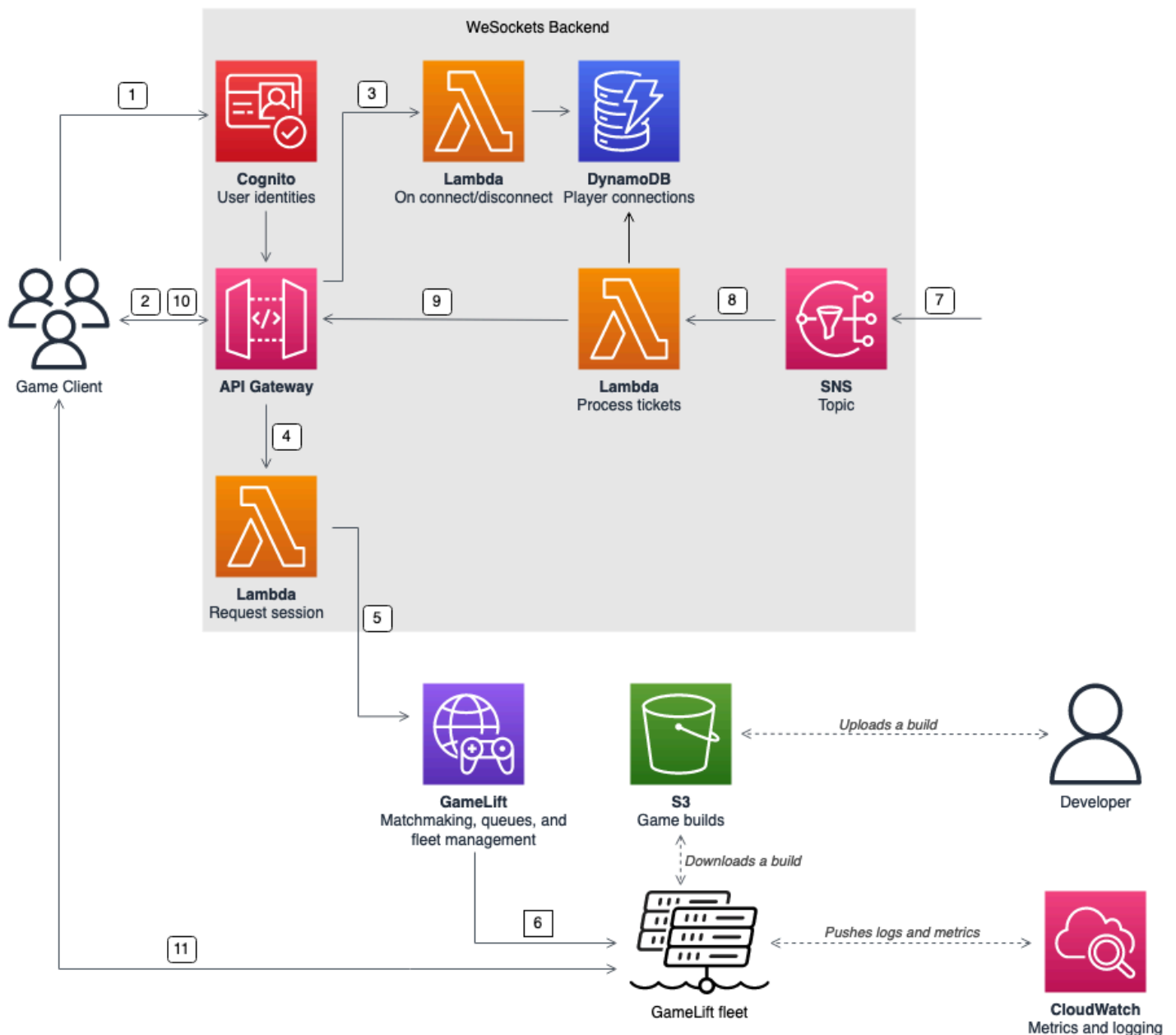
7. After Amazon GameLift Servers platziert die Sitzung an einem der Standorte der Flotte und Amazon GameLift Servers sendet eine Ereignisbenachrichtigung an ein Amazon Simple Notification Service (Amazon SNS) -Thema.
8. Eine Lambda-Funktion empfängt das Amazon SNS SNS-Ereignis und verarbeitet es.
9. Wenn das Matchmaking-Ticket ein MatchmakingSucceeded Ereignis ist, schreibt die Lambda-Funktion das Ergebnis zusammen mit dem Port und der IP-Adresse des Spieleservers in eine DynamoDB-Tabelle.
- 10Der Spielclient sendet eine signierte Anfrage an API Gateway, um den Status des Matchmaking-Tickets in einem bestimmten Intervall einzusehen.
- 11API Gateway verwendet eine Lambda-Funktion, die den Status des Matchmaking-Tickets überprüft.
- 12Die Lambda-Funktion überprüft die DynamoDB-Tabelle, um festzustellen, ob das Ticket erfolgreich ist. Wenn sie erfolgreich war, sendet die Funktion den Port und die IP-Adresse des Spieleservers zusammen mit der Sitzungs-ID des Spielers zurück an den Client. Wenn das Ticket nicht erfolgreich war, sendet die Funktion eine Antwort, in der bestätigt wird, dass das Spiel noch nicht bereit ist.
- 13Der Spielclient stellt über TCP oder UDP eine Verbindung zum Spieleserver her, indem er den Port und die IP-Adresse verwendet, die der Backend-Dienst bereitstellt. Der Spielclient sendet dann die Sitzungs-ID des Spielers an den Spieleserver, der die ID dann mithilfe des Server-SDK für validiert. Amazon GameLift Servers

Eigenständige Spielsitzungsserver mit einem WebSocket basierten Backend

Mithilfe einer auf Amazon API Gateway WebSocket basierenden Architektur können Sie mithilfe serverinitiierten Nachrichten Matchmaking-Anfragen stellen WebSockets und Push-Benachrichtigungen für den Abschluss des Matchmakings senden. Diese Architektur verbessert die Leistung durch eine bidirektionale Kommunikation zwischen dem Client und dem Server.

Weitere Informationen zur Verwendung von API Gateway WebSock APIs finden Sie unter [Arbeiten mit WebSocket APIs](#).

Das folgende Diagramm zeigt eine WebSocket basierte Backend-Architektur, die API Gateway und andere verwendet, um Spielern Spiele AWS-Services zuzuordnen, die auf Amazon GameLift Servers Flotten laufen. Die folgende Liste enthält eine Beschreibung für jeden nummerierten Callout im Diagramm.



1. Der Spielclient fordert eine Amazon Cognito Cognito-Benutzeridentität aus einem Amazon Cognito Cognito-Identitätspool an.
2. Der Spielclient signiert eine WebSocket Verbindung zu einer API-Gateway-API mit den Amazon Cognito Cognito-Anmeldeinformationen.
3. API Gateway ruft eine AWS Lambda Funktion für die Verbindung auf. Die Funktion speichert die Verbindungsinformationen in einer Amazon DynamoDB-Tabelle.
4. Der Spielclient sendet über die API Gateway über die WebSocket Verbindung eine Nachricht an eine Lambda-Funktion, um eine Sitzung anzufordern.

5. Eine Lambda-Funktion empfängt die Nachricht und fordert dann über Amazon GameLift Servers FlexMatch Matchmaking eine Übereinstimmung an.
6. Nach FlexMatch Spielen FlexMatch fordert eine Gruppe von Spielern über eine Warteschlange eine Platzierung für eine Spielsitzung an Amazon GameLift Servers.
7. After Amazon GameLift Servers platziert die Sitzung an einem der Standorte der Flotte und Amazon GameLift Servers sendet eine Ereignisbenachrichtigung an ein Amazon Simple Notification Service (Amazon SNS) -Thema.
8. Eine Lambda-Funktion empfängt das Amazon SNS SNS-Ereignis und verarbeitet es.
9. Wenn es sich bei dem Matchmaking-Ticket um ein MatchmakingSucceeded Ereignis handelt, fordert die Lambda-Funktion die richtige Spielerverbindung von DynamoDB an. Die Funktion sendet dann über die API Gateway über die WebSocket Verbindung eine Nachricht an den Spielclient. In dieser Architektur fragt der Spielclient den Status der Spielersuche nicht aktiv ab.
10. Der Spielclient erhält über die WebSocket Verbindung den Port und die IP-Adresse des Spielerservers sowie die Sitzungs-ID des Spielers.
11. Der Spielclient stellt über TCP oder UDP eine Verbindung zum Spieleserver her und verwendet dabei den Port und die IP-Adresse, die der Backend-Dienst bereitstellt. Der Spielclient sendet außerdem die Sitzungs-ID des Spielers an den Spieleserver, der die ID dann mithilfe des Server-SDK für validiert. Amazon GameLift Servers

Stellen Sie Hosting-Flotten bereit für Amazon GameLift Servers

Die Bereitstellung von Hosting-Ressourcen umfasst die Erstellung und Konfiguration der Computerinfrastruktur, auf der Ihre Spieleserver ausgeführt werden. Amazon GameLift Servers bietet verschiedene Flottenarten, um unterschiedlichen Hosting-Anforderungen gerecht zu werden, von vollständig verwalteten AWS Cloud Ressourcen bis hin zu Hybridlösungen, die Cloud- und lokale Infrastruktur kombinieren.

Wählen Sie den Flottentyp, der Ihren Anforderungen in Bezug auf Kosten, Kontrolle, Skalierbarkeit und geografische Verteilung am besten entspricht. Sie können auch mehrere Flottenarten in einer einzigen Hosting-Lösung kombinieren, um sie für unterschiedliche Szenarien oder Spielerpopulationen zu optimieren.

Eigenschaften der Flotte

Eine Amazon GameLift Servers Flotte ist eine Sammlung von Computerressourcen, die deine Spieleserver betreiben und Spielsitzungen für Spieler veranstalten. Flotten können sich in der

Art der verwendeten Rechenressourcen und in der Art und Weise, wie die Flotte verwaltet wird, unterscheiden. Die Größe einer Flotte — die Anzahl der Spielsitzungen und Spieler, die sie unterstützen kann — hängt von der Anzahl der Rechenressourcen ab, die Sie ihr zur Verfügung stellen. Alle Amazon GameLift Servers Flotten haben die folgenden Eigenschaften:

- Die Spielserverprozesse, die auf allen Flotten ausgeführt werden, sind in das Server-SDK für den Dienst integriert Amazon GameLift Servers und kommunizieren auf dieselbe Weise mit dem Amazon GameLift Servers Dienst. Spieleserver melden ihre Verfügbarkeit, um Spielsitzungen und Spieler zu veranstalten, auf Aufforderungen zum Starten oder Beenden von Spielsitzungen und anderen Interaktionen zu reagieren.
- Amazon GameLift Servers verwaltet die Platzierung von Spielsitzungen für alle Flotten auf die gleiche Weise. Amazon GameLift Servers verfolgt den Spielserverstatus einer Flotte und wählt aus verfügbaren Spielservern aus, um eine neue Spielsitzung abzuhalten. Dieser Prozess wird verwendet, unabhängig davon, ob dein Spiel Spielsitzungen auf einer einzigen Flotte platziert oder eine [Warteschlange für Spielsitzungen](#) verwendet, um das Hosting auf mehrere Flotten zu verteilen. Mit einer Warteschlange kannst du deine Platzierungsentscheidungen auch anpassen, um Faktoren wie Ressourcenkosten und Latenz zu berücksichtigen.
- Alle Flotten unterstützen den Einsatz eines FlexMatch Matchmakers in Zusammenarbeit mit einer Warteschlange für die Platzierung von Spielsitzungen. Der Amazon GameLift Servers Dienst empfängt Spielanfragen von Spielern, erstellt die Spiele und leitet sie an die Warteschlange für Spielsitzungen weiter, um nach verfügbaren Spielservern zu suchen.
- Amazon GameLift Servers sammelt eine Vielzahl von Flottenkennzahlen. Dazu gehören Statusmetriken für Computer und Serverprozesse sowie Nutzungsmetriken für Spielsitzungen und Spieleraktivitäten. Die vollständige Liste der verfügbaren Messwerte finden Sie unter [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#).

Themen

- [Entscheidungshilfe: Wählen Sie eine Hosting-Option](#)
- [Amazon GameLift Servers verwaltete EC2 Flotten](#)
- [Amazon GameLift Servers verwaltete Containerflotten](#)
- [Amazon GameLift Servers Überall Flotten](#)
- [Erstellen Sie eine Hybrid-Hosting-Lösung](#)

Entscheidungshilfe: Wählen Sie eine Hosting-Option

Amazon GameLift Servers bietet mehrere Hosting-Optionen, die jeweils für unterschiedliche Hosting-Szenarien und -Anforderungen konzipiert sind. Dieser Leitfaden hilft dir bei der Auswahl des Flottentyps, der den technischen Anforderungen, den betrieblichen Präferenzen und den geschäftlichen Einschränkungen deines Spiels entspricht. Allgemeine Beschreibungen finden Sie unter [Amazon GameLift Servers Hosting-Optionen für Spiele](#).

Vergleich von Flotten

In der folgenden Tabelle werden die wichtigsten Merkmale der einzelnen Hosting-Optionen verglichen:

Hosting-Option	Am besten für	Führungsebene	Wichtigste Vorteile
Verwaltet EC2	Produktions-Hosting mit vollständiger AWS Verwaltung	Vollständig verwaltet	Automatische Skalierung, globale Reichweite, minimaler betrieblicher Aufwand
Verwaltete Container	Containerisierte Anwendungen, die Orchestrierung erfordern	Vollständig verwaltet	Container-Orchestrierung, Ressourceneffizienz, moderne Bereitstellung
Überall	Lokale, Hybrid- oder Entwicklungsumgebungen	Selbstverwaltet	Hardwarekontrolle, Compliance-Anforderungen, Kostenoptimierung

Wichtige Entscheidungsfaktoren

Berücksichtigen Sie diese Faktoren bei der Auswahl Ihrer Hosting-Option:

Operative Komplexität

Wie viel Infrastrukturmanagement möchten Sie bewältigen? Verwaltete Flotten erfordern nur minimalen Betriebsaufwand, während Anywhere-Flotten Ihnen die volle Kontrolle bieten, aber mehr Management erfordern.

Anforderungen an die Skalierbarkeit

Benötigen Sie eine automatische Skalierung auf der Grundlage der Spielernachfrage?

Verwaltete Flotten bieten eine integrierte auto-scaling, während Anywhere-Flotten ein manuelles Kapazitätsmanagement erfordern.

Geografische Verteilung

Wo befinden sich Ihre Spieler? Verwaltete Flotten können in mehreren AWS Regionen weltweit eingesetzt werden, wohingegen Anywhere-Flotten von Ihren eigenen Infrastrukturstandorten abhängen.

Compliance und Sicherheit

Haben Sie spezielle Compliance-Anforderungen oder müssen Sie Daten lokal speichern? Mit Anywhere-Flotten behalten Sie die volle Kontrolle über Ihre Infrastruktur und Ihren Datenstandort.

Kostenüberlegungen

Was ist Ihre Strategie zur Budget- und Kostenoptimierung? Berücksichtigen Sie beim Vergleich der Optionen sowohl die Infrastrukturkosten als auch die Betriebskosten.

Hybride Lösungen

Sie müssen sich nicht nur für eine Art von Flotte für Ihre Game-Hosting-Lösung entscheiden. Viele Spiele verwenden einen hybriden Ansatz, bei dem mehrere Flottenarten kombiniert werden, um sie für verschiedene Szenarien zu optimieren:

- **Entwicklung und Testen:** Verwenden Sie Anywhere-Flotten für Entwicklung und Tests und stellen Sie sie dann für die Produktion auf verwalteten Flotten bereit.
- **Regionale Optimierung:** Verwenden Sie verwaltete Flotten AWS-Regionen mit hoher Spielerdichte und Anywhere-Flotten in Regionen, in denen Sie über eine bestehende Infrastruktur verfügen.
- **Kostenoptimierung:** Verwenden Sie verwaltete Flotten für die Basiskapazität und Anywhere-Flotten für Überlastungen oder spezifische Workloads.

Amazon GameLift Serversverwaltete EC2 Flotten

Amazon GameLift Serversverwaltete EC2 Flotten bieten Cloud-basierte Ressourcen für das Produktionshosting. Mit einer verwalteten Flotte erhalten Sie die Flexibilität, Sicherheit und

Zuverlässigkeit von AWS Cloud Ressourcen, die für das Hosten von Spielen im Mehrspielermodus optimiert sind. Amazon GameLift Servers bietet robuste Tools zur Hostverwaltung.

Eine verwaltete EC2 Flotte besteht aus einer Reihe von Amazon Elastic Compute Cloud (Amazon EC2) -Instances, die Ihre Konfiguration Amazon GameLift Servers besitzen und auf Grundlage Ihrer Konfiguration betrieben werden. Diese Instances befinden sich physisch in unterstützten AWS-Regionen oder Local Zones. Wenn du eine Flotte erstellst, wählst du einen EC2 Instance-Typ, der die Anforderungen deines Spieleservers an Rechenleistung, Arbeitsspeicher, Speicherplatz und Netzwerkfähigkeit erfüllt.

Beim Start jeder Instanz in der Flotte Amazon GameLift Servers wird dein Gameserver-Build mit der erforderlichen Laufzeitumgebung bereitgestellt. Die Laufzeitumgebung verwendet die neueste Amazon Machine Image (AMI) -Version, die bei der Erstellung der Flotte verfügbar war. Alle Instances in der Flotte verwenden dieselbe AMI-Version.

Note

Als bewährte Methode empfehlen wir, Ihre Flotten alle 30 Tage auszutauschen, um eine sichere up-to-date Laufzeitumgebung für Ihre gehosteten Spieleserver aufrechtzuerhalten. Dazu musst du eine neue Flotte erstellen und den Spielerverkehr dorthin migrieren. Weitere Anleitungen finden Sie unter [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#).

Nach der Installation der Laufzeitumgebung und des Builds Ihres Spieleservers auf einer Instanz werden die Spieleserverprozesse Amazon GameLift Servers gestartet. Jeder Spieleserverprozess stellt eine Verbindung zum Amazon GameLift Servers Dienst her, meldet die Bereitschaft, eine Spielsitzung zu veranstalten, und beginnt mit der Übermittlung des Gesundheitsstatus. Amazon GameLift Servers kann dann den Serverprozess auffordern, eine Spielsitzung zu starten.

Übernimmt zusätzlich zur Amazon GameLift Servers Flottenbereitstellung die folgenden Hostverwaltungsaufgaben, sodass Sie sich nicht darum kümmern müssen:

- Verfolgt den Status aller Computer in der Flotte und ersetzt veraltete oder fehlerhafte Berechnungen.
- Verwaltet die Authentifizierung für die Kommunikation zwischen Serverprozessen und dem Dienst. Amazon GameLift Servers

- Startet und stoppt automatisch Spielserverprozesse auf jedem Computer, basierend auf Ihrer Laufzeitkonfiguration.
- Bietet Tools zur auto Skalierung, um die Flottenkapazität dynamisch an die Nachfrage der Spieler anzupassen.
- Meldet Leistungskennzahlen für die EC2 Instanzen der Flotte.

In den folgenden Themen erfahren Sie, wie Sie verwaltete EC2 Flotten einrichten und verwalten können:

- [Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwaltetem EC2](#)
- [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#)
- [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#)
- [Anpassungen der Hosting-Ressourcen](#)
- [Eine Amazon GameLift Servers Flottenkonfiguration aktualisieren](#)

Workflow zur Erstellung verwalteter EC2 Flotten

Für verwaltete Flotten Amazon GameLift Servers richtet er die Flottenressource ein und stellt außerdem eine Reihe von Rechenressourcen bereit, während deine Gameserver-Software installiert ist und läuft. Wenn der Erstellungsprozess abgeschlossen und erfolgreich ist, verfügt die Flotte über eine aktive EC2 Instanz in der Fleet-Heimatregion und jeweils eine an den abgelegenen Standorten der Flotte. Alle Instanzen haben das Spiel und sind bereit, Spielsitzungen abzuhalten.

1. Amazon GameLift Servers erstellt die Flottenressource in der Heimatregion der Flotte und legt die gewünschte Kapazität an jedem Standort auf eine (1) Instanz fest. Der Status der Flotte und des Standorts ist auf Neu gesetzt.
2. Amazon GameLift Servers beginnt mit dem Schreiben von Ereignissen in das Flottenereignisprotokoll.
3. Amazon GameLift Servers setzt den Flottenstatus auf Wird heruntergeladen und beginnt, die Spielserver-Software für den Einsatz vorzubereiten.
 - a. Ruft den hochgeladenen Spielserver-Build ab und extrahiert die komprimierten Dateien.
 - b. Führt Installationsskripte aus, falls vorhanden.
 - c. Setzt den Flottenstatus auf Validieren und beginnt mit der Überprüfung, ob beim Herunterladen und Installieren der Build-Dateien keine Fehler aufgetreten sind.

4. Amazon GameLift Servers setzt den Flottenstatus auf Building, konfiguriert die Flottenhardware und weist jeder EC2 Flotteninstanz eine Instanz zu.
5. Amazon GameLift Servers setzt den Flottenstatus auf Aktiviert. Startet auf jeder Instanz einen Spielservers-Prozess (basierend auf den Runtime-Anweisungen der Flotte) und testet die Konnektivität zwischen dem Build und dem Amazon GameLift Servers Service.
6. Wenn die Spielserversprozesse auf jeder Instanz eine Verbindung herstellen und die Bereitschaft zum Hosten von Spielsitzungen melden, Amazon GameLift Servers wird der Status der Flotte und des Standorts auf Aktiv gesetzt. Zu diesem Zeitpunkt gilt die Flotte als bereit, Spielsitzungen abzuhalten.

Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte

In diesem Thema wird beschrieben, wie Sie eine Amazon GameLift Servers verwaltete EC2 Flotte erstellen. Verwaltete Flotten verwenden Amazon Elastic Compute Cloud (Amazon EC2) Compute Instances, die für das Hosten von Spielen im Mehrspielermodus optimiert sind. Sie können verwaltete Flotten erstellen, die Computer in globalen AWS-Regionen und Local Zones bereitstellen, die von unterstützt werden. Amazon GameLift Servers

Wenn Sie eine neue verwaltete EC2 Flottenressource erstellen, leiten Sie sofort die erste Phase der Flottenerstellung ein. Die verwaltete Flottenerstellung durchläuft mehrere Phasen Amazon GameLift Servers. Dabei wird eine EC2 Instanz bereitgestellt, eine Laufzeitumgebung installiert und Ihr Spielservers baut auf der Instanz auf und beginnt mit dem Starten der Spielservers. Abhängig von der Laufzeitumgebung, die Ihr Spielservers-Build benötigt, Amazon GameLift Servers stellt er die neueste Version des Amazon Machine Image (AMI) zum Zeitpunkt der Flottenerstellung bereit (und alle future Instances in der Flotte werden dieselbe Version verwenden). Sie können den Status einer Flotte in der Konsole oder mit AWS Command Line Interface (AWS CLI) überwachen. Wenn eine Flotte bereit ist, Spielsitzungen abzuhalten, ändert sich der Status auf. ACTIVE Weitere Informationen zur verwalteten Flottenerstellung finden Sie in den folgenden Themen:

Note

Als bewährte Methode empfehlen wir, Ihre Flotten alle 30 Tage auszutauschen, um eine sichere und up-to-date Laufzeitumgebung für Ihre gehosteten Spielservers aufrechtzuerhalten. Dazu musst du eine neue Flotte erstellen und den Spielerverkehr dorthin migrieren. Weitere Anleitungen finden Sie unter [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#).

Um eine verwaltete EC2 Flotte zu erstellen

Verwenden Sie entweder die Amazon GameLift Servers Konsole oder die AWS Command Line Interface (AWS CLI), um eine verwaltete EC2 Flotte zu erstellen.

Console

Öffnen Sie in der [Amazon GameLift ServersKonsole](#) im Navigationsbereich die Seite Verwaltet EC2, Flotten. Wählen Sie „Verwaltete EC2 Flotte erstellen“, um den Workflow zur Flottenerstellung zu starten.

Schritt 1: Definieren Sie die Details der verwalteten EC2 Flotte

 Für eine minimale Flottenkonfiguration:

- Geben Sie einen Flottennamen an.
- Wählen Sie Binärtyp: Build. Wählen Sie einen Build aus der Liste der Server-Builds aus, auf die Sie zuvor hochgeladen habenAmazon GameLift Servers.
- Überspringen Sie die Abschnitte Zusätzliche Details und Tags, um die Standardeinstellungen zu verwenden.

1. Füllen Sie den Abschnitt mit den Flottendetails aus:

- a. Geben Sie einen Flottennamen ein. Flottennamen sind nur beschreibend und nützlich, um eine Liste von Flotten zu sortieren oder zu filtern. Sie müssen nicht einzigartig sein.
- b. Geben Sie eine kurze Flottenbeschreibung an, die alle benutzerdefinierten Informationen enthält, die Sie über die Flotte erfassen möchten.
- c. Wählen Sie Binärtyp und Build aus und verwenden Sie die Drop-down-Liste, um einen Spielserver-Build auszuwählen, auf Amazon GameLift Servers den Sie zuvor hochgeladen haben.

2. Lege nach Bedarf zusätzliche Details fest.

- a. Wenn die ausführbare Datei des Spielerservers auf andere AWS Ressourcen in Ihrem Konto zugreifen muss, geben Sie eine IAM-Instanzrolle mit den erforderlichen Berechtigungen an. Weitere Informationen, einschließlich der Autorisierung anderer serverseitiger Anwendungen (z. B. CloudWatch Agenten), finden Sie unter. [Connect](#)

deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS

Ressourcen Diese Einstellung kann nicht geändert werden, nachdem Sie die Flotte erstellt haben.

Sie müssen die Rolle erstellen, bevor Sie eine Flotte erstellen, die sie verwendet. Um eine Flotte mit einer Instance-Rolle zu erstellen, muss Ihr AWS Benutzer außerdem über PassRole IAM-Berechtigungen verfügen (siehe [Beispiele für IAM-Berechtigungen für Amazon GameLift Servers](#)).

- b. Aktiviere die Option TLS-Zertifikat generieren, um Authentifizierung und Verschlüsselung für dein Spiel einzurichten. Spieleclients verwenden dieses Zertifikat, um einen Spieleserver zu authentifizieren, wenn sie eine Verbindung herstellen, und verschlüsseln die gesamte Client/Server-Kommunikation. Für jede Instanz in einer TLS-fähigen Flotte. Amazon GameLift Servers erstellt außerdem einen neuen DNS-Eintrag mit dem Zertifikat. Diese Einstellung kann nicht geändert werden, nachdem Sie die Flotte erstellt haben.
 - c. Amazon GameLift Servers gibt Metrikdaten für jede einzelne Flotte aus. Wenn Sie Metrikdaten für mehrere Flotten kombinieren möchten, geben Sie einen Metrikgruppennamen an. Verwenden Sie denselben Metrikgruppennamen für alle Flotten, für die Sie Metriken kombinieren möchten. Wird verwendet CloudWatch , um die aggregierten Metrikgruppendaten anzuzeigen.
3. Fügen Sie der Flottenressource Tags hinzu. Jeder Tag (Markierung) besteht aus einem Schlüssel und einem optionalen Wert, beides können Sie bestimmen. Weisen Sie AWS Ressourcen, die Sie kategorisieren möchten, Tags zu, z. B. nach Zweck, Eigentümer oder Umgebung. Wählen Sie für jedes Tag, das Sie hinzufügen möchten, die Option Neues Tag hinzufügen aus.
4. Wählen Sie Weiter, um den Workflow fortzusetzen.

Schritt 2: Definieren Sie die Instanzdetails

Wählen Sie in diesem Schritt die Art der Hardwareressourcen aus, die Sie für Ihre Spieleserver verwenden möchten, und geben Sie an, wo Sie sie einsetzen möchten. Wenn Sie mehrere Standorte auswählen, können Sie Ihren Spieleserver an einem größeren geografischen Standort bereitstellen, wodurch er näher an Ihren Spielern liegt und die Latenz minimiert wird. Alle EC2 Instanzen in einer Flotte müssen vom gleichen Typ sein. Nicht alle Instance-Typen sind an allen Standorten verfügbar.

 Für eine minimale Flottenkonfiguration:

- Beginnen Sie mit der Bereitstellung ausschließlich am Heimatstandort. Für die anfängliche Entwicklung und das Testen müssen Sie keine Remote-Standorte hinzufügen.
- Stellen Sie den Flottentyp auf „On-Demand“ ein. Spot-Flotten erfordern zusätzliche Einrichtungsarbeiten.
- Stellen Sie den Instanztyp auf „c5.large“ ein. Dieser häufig verwendete Instanztyp ist in allen Versionen verfügbar. AWS-Regionen Sie können Ihre Hardwarenutzung später optimieren.

1. Geben Sie unter Instanzbereitstellung die Standorte und den Typ der Flotte an.
 - a. Wählen Sie einen oder mehrere zusätzliche Standorte aus, an denen Sie Flotteninstanzen bereitstellen möchten. Diese Remote-Standorte werden dem Heimatstandort der Flotte (der vorausgewählt ist) hinzugefügt, AWS-Region an dem Sie diese Flotte erstellen. Sie können entfernte Standorte aus allen AWS-Regionen und Local Zones auswählen, die dies Amazon GameLift Servers unterstützen.

Weitere Informationen zu unterstützten Standorten, einschließlich der Verwendung eines Standorts AWS-Region , der nicht standardmäßig aktiviert ist, findest du unter [Amazon GameLift Servers Servicestandorte](#) Für verwaltetes Hosting. Sehen Sie sich auch die Amazon GameLift Servers [Kontingente](#) für Standorte pro Flotte an.
 - b. Wählen Sie, ob Sie für diese Flotte entweder On-Demand-Instances oder Spot-Instances verwenden möchten. Weitere Informationen zu Flottenarten finden Sie unter [On-Demand-Instances im Vergleich zu Spot Instances](#).
2. Wählen Sie eine Amazon EC2 Instance-Konfiguration, die Ihren Anforderungen entspricht und an allen ausgewählten Standorten verfügbar ist. Diese Liste wird auf der Grundlage Ihres aktuellen Standorts und Ihrer Auswahl an Flottenarten gefiltert. Sie können sie weiter nach anderen Faktoren wie Instance-Typ, Familie und Architektur filtern. Nachdem Sie die Flotte erstellt haben, können Sie den Instance-Typ nicht mehr ändern.

An einigen Standorten gibt es eingeschränkte Optionen für Instance-Typen. Wenn Ihr bevorzugter Instance-Typ nicht für alle Standorte verfügbar ist, wählen Sie den Wert für die Standortverfügbarkeit, um alle Details anzuzeigen. Um alle Standorte zu

berücksichtigen, müssen Sie möglicherweise separate Flotten mit unterschiedlichen Instance-Typen erstellen.

Weitere Informationen zur Auswahl eines Instance-Typs finden Sie unter [Instance-Typen](#). Weitere Informationen zu Amazon EC2 Arm-Architekturen finden Sie unter [AWS Graviton Processor](#) und [EC2Amazon-Instance-Typen](#). Eine vollständige Liste der Instance-Typen, die von unterstützt werden Amazon GameLift Servers, finden Sie in der API-Referenz für [EC2InstanceType\(\)](#)`CreateFleet()`.

 Note

Graviton Arm-Instances erfordern einen Server-Build für ein Linux-AMI. Server-SDK 5.1.1 oder neuer ist für C++ und C# erforderlich. Server-SDK 5.0 oder neuer ist für Go erforderlich. Diese Instances bieten keine out-of-the-box Unterstützung für die Mono-Installation auf Amazon Linux 2023 (AL2023) oder Amazon Linux 2 (AL2).

3. Wählen Sie Weiter, um den Workflow fortzusetzen.

Schritt 4: Laufzeit konfigurieren

Beschreiben Sie in diesem Schritt, wie die einzelnen Instances der Flotte Ihre Gameserver-Software ausführen sollen. Definieren Sie für jede ausführbare Datei, die auf einer Instanz ausgeführt werden soll, eine separate Zeile für Serverprozesse und legen Sie fest, wie viele Serverprozesse gleichzeitig ausgeführt werden sollen. Öffnen Sie Ports auf jeder Instanz, damit Spieler eine direkte Verbindung zu Spielservers herstellen können. Du kannst diese Flotteneinstellungen jederzeit aktualisieren.

 Für eine minimale Flottenkonfiguration:

- Definieren Sie ein einzelnes Server-Prozesszeilenelement für die ausführbare Datei Ihres Spielservers. Wenn auf Ihrem Spielservers andere Prozesse ausgeführt werden müssen, erstellen Sie auch für jeden dieser Prozesse eine Definition.
- Verwenden Sie die Standardanzahl gleichzeitiger Prozesse (1) für jeden Einzelposten.
- Einstellungen zur Aktivierung der Spielsitzung überspringen.
- Geben Sie eine einzelne Portnummer an.

- Überspringe die Ressourceneinstellungen für Spielsitzungen.

1. Erstellen Sie eine Runtime-Konfiguration, um anzuweisen Amazon GameLift Servers, wie Serverprozesse auf jeder Instance in der Flotte ausgeführt werden. Sie können die Laufzeitkonfiguration einer Flotte jederzeit nach der Bereitstellung ändern.
 - a. Geben Sie den Startpfad zu einer ausführbaren Datei in Ihrem Build ein. Auf Windows-Instances werden die ausführbaren Dateien des Spieleservers für den Pfad erstellt. C:\game Auf Linux-Instances sind Spieleserver für /local/game ausgelegt. Beispiele: **C:\game\MyGame\server.exe** oder **/local/game/MyGame/server.exe**.
 - b. Geben Sie optionale Startparameter ein, um sie an die ausführbare Datei Ihres Spiels zu übergeben. Beispiel: **+sv_port 33435 +start_lobby**.
 - c. Geben Sie die Anzahl der gleichzeitigen Prozesse an, die auf jeder Instanz ausgeführt werden sollen. Bei einer ausführbaren Spieleserverdatei kann jeder Prozess eine Spielsitzung hosten, sodass gleichzeitige Prozesse die Anzahl der Spielsitzungen bestimmen, die die Instanz gleichzeitig hosten kann.

Überprüfen Sie die Amazon GameLift Servers [Kontingente](#) für Serverprozesse pro Instanz. Diese Kontingente gelten für die Gesamtzahl der gleichzeitigen Prozesse für alle Konfigurationen. Wenn Sie die Flotte so konfigurieren, dass sie diese Werte überschreitet, kann die Flotte nicht aktiviert werden.
2. Verwende die Standardeinstellungen für die Aktivierung von Spielsitzungen oder passe sie an dein Spiel an. Wenn die Laufzeitkonfiguration mehrere gleichzeitige Spieleserverprozesse pro Instanz erfordert, bestimmen diese Einstellungen, wie schnell neue Spielsitzungen gestartet werden können.
 - a. Stellen Sie die maximale Anzahl gleichzeitiger Spielesitzungen ein, um die Anzahl der Spieleserver auf einer Instanz zu begrenzen, die eine neue Spielsitzung vorbereiten. Diese Einstellung ist nützlich, wenn das Starten mehrerer neuer Spielsitzungen ressourcenintensiv ist und sich auf die Leistung anderer laufender Spielsitzungen auswirken kann.
 - b. Setze das Timeout für neue Aktivierung so, dass es die maximale Zeit angibt, die eine neue Spielsitzung bis zum Abschluss der Aktivierung benötigen sollte, und melde Spieler, die bereit sind, sie zu empfangen. Amazon GameLift Servers beendet die Aktivierung einer Spielsitzung, wenn sie diesen Wert überschreitet.

3. Öffnet die EC2 Porteinstellungen, damit eingehender Datenverkehr auf Serverprozesse auf der Flotte zugreifen kann. Diese Einstellungen sind nicht erforderlich, um eine Flotte zu erstellen, aber Sie müssen sie festlegen, bevor Spieler eine Verbindung zu Spielsitzungen auf der Flotte herstellen können.

Wähle für jede Porteinstellung den Typ des Datenübertragungsprotokolls, das für die Kommunikation zwischen deinem Spielclient und dem Spieleserver verwendet werden soll. Geben Sie einen Portbereich (im Formatnnnnn[-nnnnn]) und einen IP-Adressbereich unter Verwendung der CIDR-Notation an (z. B. **0.0.0.0/0** der Zugriff für jedermann ermöglicht).

Wenn Sie mehrere nicht aufeinanderfolgende Bereiche festlegen müssen, erstellen Sie mehrere Porteinstellungen.

4. Geben Sie optionale Ressourceneinstellungen für Spielsitzungen an. Sie können diese Einstellungen jederzeit nach der Bereitstellung aktualisieren.
 - a. Aktiviere oder deaktiviere die Game Scaling-Schutzrichtlinie für alle Instances in der Flotte. Während eines Scale-Down-Events werden geschützte Flotten-Instances Amazon GameLift Servers nicht beendet, wenn sie aktive Spielsitzungen veranstalten.
 - b. Lege ein maximales Limit für die Erstellung von Ressourcen fest, wenn du die Anzahl der Spielsitzungen einschränken möchtest, die ein Spieler während eines bestimmten Zeitraums erstellen kann.
5. Wählen Sie Weiter, um den Workflow fortzusetzen.

Schritt 5 Überprüfen und erstellen

Überprüfen Sie Ihre Einstellungen, bevor Sie die Flotte erstellen. Obwohl einige Einstellungen später aktualisiert werden können (siehe[Eine Amazon GameLift Servers Flottenkonfiguration aktualisieren](#)), sind Änderungen an den folgenden Einstellungen nicht zulässig, nachdem die Flotte erstellt wurde:

- Berechnungstyp: Sie können eine verwaltete EC2 Flotte nicht in eine Anywhere-Flotte konvertieren.
- Gameserver-Binärdatei: Um ein Update für Ihren Gameserver-Build bereitzustellen, müssen Sie eine neue Flotte erstellen.
- Zusätzliche Optionen, einschließlich Instanzrolle und Generierung von TLS-Zertifikaten.

- Instanzdetails, einschließlich Flottenart (Spot oder On-Demand) und EC2 Instance-Typ.

Wenn Sie bereit sind, die neue Flotte bereitzustellen, wählen Sie Create. Amazon GameLift Servers startet sofort den Flottenaktivierungsprozess, weist eine eindeutige ID zu und versetzt die Flotte in den NEW Status. Verfolgen Sie den Fortschritt der Flotte auf der Seite „Flotten“. Rufen Sie die Detailseite der Flotte auf und wechseln Sie zur Registerkarte „Ereignisse“.

Sie können die Hosting-Kapazität einer Flotte anpassen, nachdem die Flotte den Status AKTIV erreicht hat. Amazon GameLift Servers stellt zunächst eine Flotte mit einer einzigen Instanz an jedem Flottenstandort bereit. Sie passen die Kapazität an, indem Sie zu jedem Standort Instances hinzufügen. Weitere Informationen finden Sie unter [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#).

AWS CLI

Verwenden Sie den [create-fleet](#)-Befehl, um eine Flotte von Rechenarten zu erstellen EC2. Amazon GameLift Servers erstellt die Flottenressource in Ihrer aktuellen Standard AWS-Region (oder Sie können ein --region-Tag hinzufügen, um einen anderen Wert anzugeben AWS-Region).

Erstellen Sie eine verwaltete Minimalflotte

Die folgende Beispielanforderung erstellt eine neue Flotte mit den Mindesteinstellungen, die für die Bereitstellung einer Flotte mit laufenden Spieleservern erforderlich sind, zu denen Spieleclients eine Verbindung herstellen können. Die neue Flotte hat die folgenden Eigenschaften:

- Sie spezifiziert einen Spieleserver-Build, der hochgeladen wurde Amazon GameLift Servers und sich im READY Status befindet.
- Es verwendet c5.large On-Demand-Instances mit einem Betriebssystem, das dem ausgewählten Spiel-Build entspricht.
- Es legt den Standort der Flotte AWS-Region auf diese Region fest us-west-2 und stellt Instances nur in dieser Region bereit.
- Basierend auf der Laufzeitkonfiguration führt jede Recheneinheit in der Flotte einen Spieleserverprozess aus, was bedeutet, dass auf jeder Recheneinheit jeweils nur eine Spielsitzung ausgeführt werden kann. Das Timeout für die Aktivierung von Spielsitzungen ist auf den Standardwert von 300 Sekunden festgelegt, und die Anzahl der gleichzeitigen Aktivierungen ist unbegrenzt.
- Spieler können über eine einzige Port-Einstellung von einer Verbindung zu Spieleservern herstellen. 33435

- Alle anderen Funktionen sind entweder ausgeschaltet oder verwenden die Standardeinstellungen.

```
aws gamelift create-fleet \  
  --name MinimalFleet123 \  
  --description "A basic test fleet" \  
  --region us-west-2 \  
  --ec2-instance-type c5.large \  
  --fleet-type ON_DEMAND \  
  --build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \  
  --runtime-configuration "ServerProcesses=[{LaunchPath=C:\game\Bin64.dedicated\  
\MultiplayerSampleProjectLauncher_Server.exe, ConcurrentExecutions=10}]" \  
  --ec2-inbound-permissions  
  "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
```

Erstellen Sie eine vollständig konfigurierte verwaltete Flotte

Die folgende Beispielanforderung erstellt eine Produktionsflotte mit Einstellungen für alle optionalen Funktionen. Die neue Flotte weist die folgenden Eigenschaften auf:

- Sie spezifiziert einen Spielserver-Build, der hochgeladen wurde Amazon GameLift Servers und sich im READY Status befindet.
- Es verwendet c5.large On-Demand-Instances mit dem Betriebssystem, das dem ausgewählten Spiel-Build entspricht.
- Es richtet das Zuhause der Flotte in der Heimatregion us-west-2 und AWS-Region an einem entfernten Standort ein und stellt Instances dort bereit. sa-east-1
- Basierend auf der Laufzeitkonfiguration:
 - Auf jedem Computer in der Flotte werden 10 Spielserverprozesse mit denselben Startparametern ausgeführt, was bedeutet, dass auf jedem Computer bis zu 10 Spielsitzungen gleichzeitig ausgeführt werden können.
 - Auf jedem Computer können nur zwei Spielsitzungen gleichzeitig aktiviert werden. Aktivierende Spielsitzungen müssen innerhalb von 300 Sekunden (5 Minuten) bereit sein, Spieler aufzunehmen, oder sie müssen beendet werden.
- Spieler können über einen Port im folgenden Bereich eine Verbindung zu Spieleservern herstellen 33435 to 33535.
- Es generiert ein TLS-Zertifikat für die verschlüsselte Kommunikation zwischen Spielclient und Server.

- Bei allen Spielsitzungen in der Flotte ist der Spielsitzungsschutz aktiviert.
- Einzelne Spieler sind darauf beschränkt, innerhalb von 15 Minuten drei neue Spielsitzungen zu erstellen.
- Metriken für diese Flotte sind in der Metrikgruppe enthalten `AMERfleets`, in der (in diesem Beispiel) Kennzahlen für eine Gruppe von Flotten in Nord-, Mittel- und Südamerika zusammengefasst werden.

```
aws gamelift create-fleet \
  --name ProdFleet123 \
  --description "A fully configured prod fleet" \
  --ec2-instance-type c5.large \
  --region us-west-2 \
  --locations "Location=sa-east-1" \
  --fleet-type ON_DEMAND \
  --build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \
  --certificate-configuration "CertificateType=GENERATED" \
  --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
MaxConcurrentGameSessionActivations=2, ServerProcesses=[{LaunchPath=C:\game
\Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe, Parameters=+sv_port
33435 +start_lobby, ConcurrentExecutions=10}]" \
  --new-game-session-protection-policy "FullProtection" \
  --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
PolicyPeriodInMinutes=15" \
  --ec2-inbound-permissions
  "FromPort=33435,ToPort=33535,IpRange=0.0.0.0/0,Protocol=UDP" \
  --metric-groups "AMERfleets"
```

Wenn die Anfrage „Flotte erstellen“ erfolgreich ist, wird ein Satz von Flottenattributen Amazon GameLift Servers zurückgegeben, der die von Ihnen angeforderten Konfigurationseinstellungen und eine neue Flotten-ID enthält. Amazon GameLift Servers leitet dann den Flottenaktivierungsprozess ein und setzt den Flottenstatus und den Standortstatus auf Neu. Sie können den Status der Flotte nachverfolgen und andere Informationen zu der Flotte über die folgenden CLI-Befehle anzeigen:

- [describe-fleet-events](#)
- [describe-fleet-attributes](#)
- [describe-fleet-capacity](#)
- [describe-fleet-port-settings](#)

- [describe-fleet-utilization](#)
- [describe-runtime-configuration](#)
- [describe-fleet-location-attributes](#)
- [describe-fleet-location-capacity](#)
- [describe-fleet-location-utilization](#)

Mit diesen Befehlen können Sie die Kapazität der Flotte und andere Konfigurationseinstellungen nach Bedarf ändern:

- [update-fleet-attributes](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [update-runtime-configuration](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Amazon GameLift Serversverwaltete Containerflotten

Amazon GameLift Serversverwaltete Containerflotten bieten Cloud-basierte Ressourcen für das Hosten Ihrer containerisierten Gameserver-Software. Mit einer verwalteten Containerflotte erhalten Sie die Flexibilität, Sicherheit und Zuverlässigkeit von AWS Cloud Ressourcen, die für das Hosten von Spielen im Mehrspielermodus optimiert sind. Amazon GameLift Servers bietet robuste Tools zur Hostverwaltung.

 Beschleunigen Sie das Onboarding mit diesen Tools für verwaltete Container:

- Das [Container-Starterkit](#) optimiert die Integration und die Einrichtung der Flotte. Es erweitert deinen Spieleserver um wichtige Funktionen zur Verwaltung von Spielsitzungen und verwendet vorkonfigurierte Vorlagen, um eine Containerflotte und eine automatisierte Bereitstellungspipeline für deinen Spieleserver aufzubauen. Verwende nach der Bereitstellung die Amazon GameLift Servers Konsole und die API-Tools, um die Leistung deiner Flotte zu überwachen, Spielsitzungen zu verwalten und Messwerte zu analysieren.
- Entwickler von Unreal Engine oder Unity können die Game [Amazon GameLift ServersEngine-Plug-ins und den Server](#) verwenden, SDKs um ihren Spieleserver zu

integrieren und eine Containerflotte aus der Entwicklungsumgebung Ihrer Game Engine heraus aufzubauen. Die geführten Workflows des Plugins helfen dir dabei, eine schnelle, einfache Lösung mit cloudbasiertem Hosting mithilfe verwalteter Container zu erstellen. Du kannst auf dieser Grundlage aufbauen, um eine maßgeschneiderte Hosting-Lösung für dein Spiel zu erstellen.

Eine verwaltete Containerflotte besteht aus einer Reihe von Amazon Elastic Compute Cloud (Amazon EC2) -Instances, auf denen Linux ausgeführt wird, das Amazon GameLift Servers Eigentum und der Betrieb auf der Grundlage Ihrer Konfiguration ist. Diese Instances befinden sich physisch in unterstützten AWS-Regionen oder Local Zones. Wenn du eine Containerflotte erstellst, wählst du einen EC2 Instance-Typ, der die Anforderungen deines Gameservers an Rechenleistung, Arbeitsspeicher, Speicherplatz und Netzwerkfähigkeit erfüllt.

Für eine verwaltete Containerflotte speichern Sie Linux-basierte Container-Images in einem Amazon Elastic Container Registry (Amazon ECR) -Repository und erstellen eine Container-Gruppendefinition, um Ihre Container-Architektur zu beschreiben. Wenn Sie eine Flotte erstellen, stellt Amazon GameLift Servers sie eine Flotteninstanz mit der neuesten Version des Linux Amazon Machine Image (AMI) bereit und verwendet die Container-Gruppendefinition, um Ihre Container-Images bereitzustellen. Alle Instances in einer Containerflotte verwenden dieselbe AMI-Version, auch wenn Sie eine Container-Gruppendefinition aktualisieren oder ein Container-Image ändern.

Note

Als bewährte Methode empfehlen wir, Ihre Flotten alle 30 Tage auszutauschen, um eine sichere up-to-date Laufzeitumgebung für Ihre gehosteten Spieleserver aufrechtzuerhalten. Dazu musst du eine neue Flotte erstellen und den Spielerverkehr dorthin migrieren. Weitere Anleitungen finden Sie unter [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#).

Nach der Bereitstellung der containerisierten Instance beginnen Container damit, Spieleserverprozesse zu starten. Jeder Spieleserver-Prozess stellt eine Verbindung zum Amazon GameLift Servers Dienst her, meldet die Bereitschaft, eine Spielsitzung zu veranstalten, und beginnt mit der Übermittlung des Gesundheitsstatus. Amazon GameLift Servers kann dann den Serverprozess auffordern, eine Spielsitzung zu starten.

Erledigt neben der Flottenbereitstellung auch die folgenden Aufgaben zur Hostverwaltung, sodass Sie sich nicht darum kümmern müssen: Amazon GameLift Servers

- Verfolgt den Status aller Container in der Flotte und ersetzt veraltete oder fehlerhafte Container.
- Verwaltet die Authentifizierung für die Kommunikation zwischen Serverprozessen und dem Amazon GameLift Servers Dienst.
- Bietet Tools zur auto Skalierung, die die Flottenkapazität dynamisch an die Nachfrage der Spieler anpassen.
- Meldet Leistungskennzahlen für die EC2 Instances, Container und Serverprozesse der Flotte.

In den folgenden Themen erfahren Sie, wie Sie verwaltete Containerflotten einrichten und verwalten können:

- [Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwalteten Containern](#)
- [Erstellen Sie eine Amazon GameLift Servers verwaltete Containerflotte](#)
- [Passen Sie eine Amazon GameLift Servers Containerflotte an](#)
- [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#)
- [Aktualisieren Sie eine Amazon GameLift Servers verwaltete Containerflotte](#)

So funktionieren Container in Amazon GameLift Servers

Amazon GameLift Servers Container-Flotten sind so konzipiert, dass sie Ihnen Flexibilität bei der Bereitstellung und Skalierung Ihrer containerisierten Anwendungen bieten. Es verwendet den Amazon Elastic Container Service (Amazon ECS), um die Bereitstellung und Ausführung von Aufgaben für Ihre Amazon GameLift Servers Flotten zu verwalten. In diesem Thema werden die grundlegenden Strukturelemente für den Betrieb von Containern auf einer Amazon GameLift Servers verwalteten Flotte beschrieben, gängige Architekturen veranschaulicht und einige Kernkonzepte skizziert.

 Beschleunigen Sie das Onboarding mit diesen Tools für verwaltete Container:

- Das [Container-Starterkit](#) optimiert die Integration und die Einrichtung der Flotte. Es erweitert deinen Spieleserver um wichtige Funktionen zur Verwaltung von Spielsitzungen und verwendet vorkonfigurierte Vorlagen, um eine Containerflotte und eine automatisierte Bereitstellungspipeline für deinen Spieleserver aufzubauen. Verwende nach der

Bereitstellung der Amazon GameLift Servers Konsole und die API-Tools, um die Leistung deiner Flotte zu überwachen, Spielsitzungen zu verwalten und Messwerte zu analysieren.

- Entwickler von Unreal Engine oder Unity können die [Amazon GameLift ServersPlugins](#) verwenden, um ihren Spieleserver zu integrieren und eine Containerflotte aus der Entwicklungsumgebung Ihrer Game Engine heraus aufzubauen. Die geführten Workflows des Plugins helfen dir dabei, eine schnelle, einfache Lösung mit cloudbasiertem Hosting mithilfe verwalteter Container zu erstellen. Dann baue auf dieser Grundlage auf, um eine maßgeschneiderte Hosting-Lösung für dein Spiel zu erstellen.

Komponenten der Containerflotte

Flotte

Eine Containerflotte ist eine Sammlung von Amazon EC2 EC2-Instances zum Hosten Ihrer containerisierten Spieleserver. Diese Instances werden in Amazon GameLift Servers Ihrem Namen verwaltet. Wenn Sie eine Flotte erstellen, konfigurieren Sie, wie die Container-Architektur mit Ihrer Gameserver-Software für jede Flotteninstanz bereitgestellt wird. Sie können eine Containerflotte mit Instances an einem oder mehreren geografischen Standorten erstellen. Sie können Amazon GameLift Servers Skalierungstools verwenden, um die Kapazität einer Containerflotte für Spielsitzungen und Spieler automatisch zu skalieren.

Instance

Eine Amazon EC2 EC2-Instance ist der virtuelle Server, der Rechenkapazität für das Hosting von Spielen bereitstellt. Mit Amazon GameLift Servers können Sie aus einer Reihe von Instance-Typen wählen. Jeder Instance-Typ bietet eine andere Kombination aus CPU-, Arbeitsspeicher-, Speicher- und Netzwerkkapazität.

Wenn Sie eine Containerflotte erstellen, werden Amazon GameLift Servers Ihre Container auf der Grundlage des von Ihnen ausgewählten Instance-Typs und Ihrer Flottenkonfiguration bereitgestellt. Jede bereitgestellte Flotteninstanz ist identisch und führt deine containerisierte Gameserver-Software auf die gleiche Weise aus. Die Anzahl der Instanzen in einer Flotte bestimmt die Größe der Flotte und die Hosting-Kapazität für Spiele.

Container-Gruppe

Amazon GameLift Servers verwendet das Konzept einer Containergruppe, um eine Reihe von Containern zu beschreiben und zu verwalten. Eine Container-Gruppe ähnelt einer Container-"Task" oder einem „Pod“. Innerhalb jeder Containergruppe können Sie

definieren, wie sich Container verhalten, Abhängigkeiten einrichten und verfügbare CPU- und Speicherressourcen gemeinsam nutzen.

Jede Flotteninstanz kann die folgenden Arten von Containergruppen haben:

- Eine Spielserver-Containergruppe verwaltet die Container, auf denen Ihre Spieleserver-Anwendung und unterstützende Software ausgeführt werden. Eine Containerflotte muss über eine dieser Containergruppen verfügen, um Spielsitzungen und Spieler abzuhalten. Eine Spielserver-Containergruppe kann innerhalb einer Flotteninstanz repliziert werden. Die Anzahl der Spielservergruppen-Replikate pro Flotteninstanz hängt von den Rechenanforderungen Ihrer Software und den auf der Instance verfügbaren Rechenressourcen ab.
- Eine Containergruppe pro Instanz, die optional ist, gibt Ihnen die Möglichkeit, zusätzliche Software auf jeder Flotteninstanz auszuführen. Sie sind nützlich für die Ausführung von Hintergrunddiensten oder Hilfsprogrammen, z. B. für die Überwachung. Ihre Gameserver-Software hängt nicht direkt von Prozessen in einer Instanzgruppe ab. Für jede Flotteninstanz wird nur eine Kopie einer Containergruppe pro Instanz bereitgestellt.

Jede Containergruppe in einer Containerflotte hat einen Container, der als „unverzichtbar“ gekennzeichnet ist. Ein unverzichtbarer Container bestimmt den Lebenszyklus einer Containergruppe. Wenn der Essential-Container ausfällt, wird die gesamte Containergruppe neu gestartet.

Container

Der Container ist das grundlegendste Element einer containerbasierten Architektur. Er enthält ein Container-Image mit ausführbaren Softwaredateien und abhängigen Dateien. Definieren Sie einen Container, um zu konfigurieren, wie die Software ausgeführt wird und mit ihnen interagiert.

Amazon GameLift Servers

Amazon GameLift Servers definiert zwei Arten von Containern:

- Ein Gameserver-Container enthält alles, was Sie zum Ausführen Ihrer Gameserver-Prozesse und zum Hosten von Spielsitzungen für Spieler benötigen. Er beinhaltet den Build deines Gameservers und die dazugehörige Software. Definiere einen Gameserver-Container für die Gameserver-Containergruppe einer Flotte. Der Spielserver-Container wird automatisch als unverzichtbar für die Containergruppe angesehen.
- In einem Support-Container wird zusätzliche Software zur Unterstützung deines Spieleservers ausgeführt. Es ähnelt dem Konzept eines „Sidecar“-Containers. Er bietet dir die Möglichkeit, unterstützende Software neben deinen Spieleservern auszuführen und zu skalieren, sie aber als separate Container zu verwalten. In einer Spieleserver-Containergruppe können

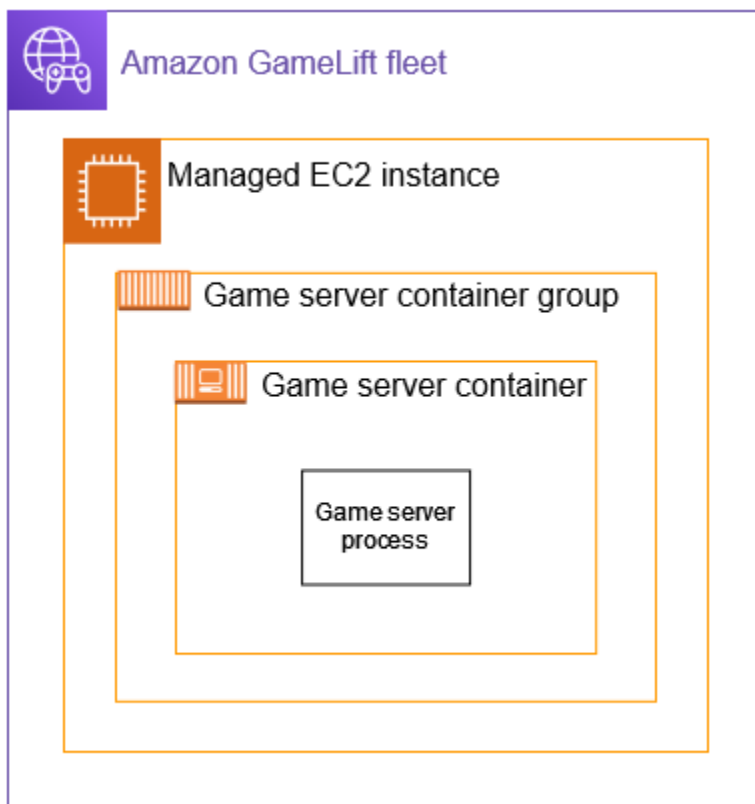
Sie keine oder mehr Support-Container definieren. In einer Containergruppe pro Instanz sind alle Container Support-Container. Jeder Support-Container kann als unverzichtbar eingestuft werden.

Datenverarbeitung

Ein Compute stellt eine Kopie einer Spielservers-Containergruppe auf einer Flotteninstanz dar.

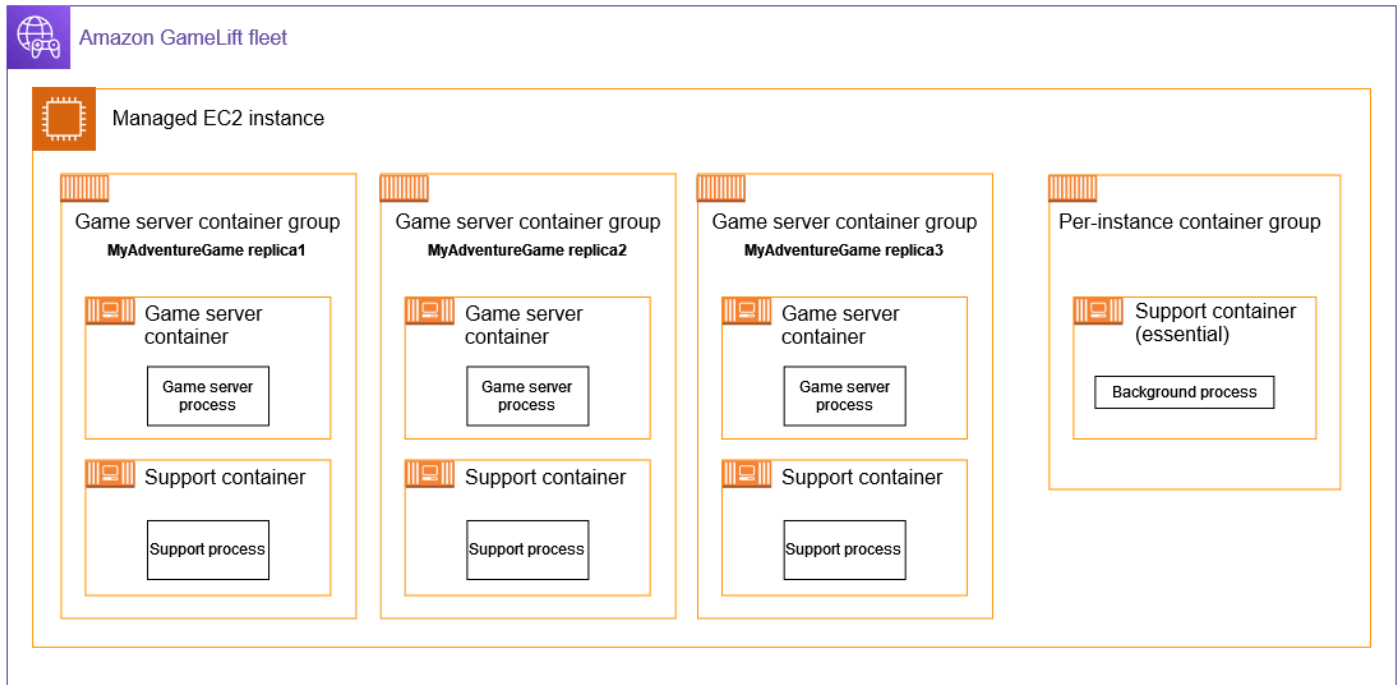
Gängige Architekturen

Das folgende Diagramm zeigt die einfachste Struktur der Containerflotte. In dieser Struktur verwaltet jede Instanz in der Flotte eine Kopie der Spielservers-Containergruppe. Die Container-Gruppe hat einen einzelnen Spieleserver-Container, auf dem ein Spieleserver-Prozess ausgeführt wird. In diesem Beispiel ist die Containerflotte so konfiguriert, dass pro Instanz eine Kopie der Spielservers-Containergruppe platziert wird. Bei dieser Architektur führt jede Instanz einen Gameserver-Prozess aus.



Dieses zweite Diagramm veranschaulicht eine komplexere Containerflottenarchitektur. In dieser Struktur besteht die Flotte sowohl aus einer Spielservers-Containergruppe als auch aus einer Containergruppe pro Instanz. Die Spielservers-Containergruppe hat separate Container für den

Spieler-Server-Prozess und einen Support-Prozess. Die Flotte ist so konfiguriert, dass drei Kopien der Spieler-Server-Containergruppe auf jeder Flotteninstanz platziert werden. Die Containergruppe pro Instanz wird niemals repliziert. In diesem Beispiel ist die Containerflotte so konfiguriert, dass drei Kopien der Spieler-Server-Containergruppe pro Instanz platziert werden. Bei dieser Architektur führt jede Instanz drei Spieler-Server-Prozesse aus.



Kern-Features

In diesem Abschnitt wird zusammengefasst, wie einige grundlegende Container-Konzepte Amazon GameLift Servers implementiert werden. Anweisungen zur Arbeit mit Containerflotten finden Sie in den entsprechenden Themen in diesem Handbuch.

Aktive Flottenupdates

Managed Containers bietet erweiterten Support, der Sie bei der Verwaltung des Lebenszyklus Ihrer gehosteten Software und Container-Architektur unterstützt. Sie können Ihre Containerdefinitionen, einschließlich Container-Images, aktualisieren und Änderungen an Ihren vorhandenen Flotten vornehmen. Diese Funktion macht es schneller und einfacher, Änderungen an Ihren Containern während der Entwicklung zu wiederholen. Es bietet auch Funktionen, mit denen Sie Ihre Softwareversionsupdates erstellen, bereitstellen und im Laufe der Zeit verfolgen können. Diese Features umfassen:

- Verwalten Sie Definitionsupdates und Versionsverwaltung für Containergruppen. Sie können fast alle Eigenschaften einer Container-Gruppendefinition aktualisieren, einschließlich Container-Images und Konfigurationseinstellungen. Jedes Mal, wenn Sie einen Container aktualisieren, weist er dem Update Amazon GameLift Servers automatisch eine Versionsnummer zu und behält standardmäßig alle Versionen bei. Sie können auf jede bestimmte Version zugreifen und Versionen nach Bedarf löschen. Beim Erstellen einer Containerflotte können Sie die Definition und Version der Containergruppe angeben, die bereitgestellt werden soll.
- Aktualisieren Sie bestehende Containerflotten mit neuen Containergruppendefinitionen und Konfigurationseinstellungen. Sie können Container-Updates für Flotten bereitstellen, die bereits für Flotteninstanzen bereitgestellt wurden. Sie können den Status der Aktualisierungsbereitstellungen an jedem Flottenstandort mithilfe der AWS-Managementkonsole, der AWS SDK und der CLI verfolgen.
- Konfigurieren Sie, wie Flottenupdates für eine aktive Flotte bereitgestellt werden sollen.
 - Schutz von Spielsitzungen. Entscheiden Sie sich dafür, Flotteninstanzen mit aktiven Spielsitzungen bis zum Ende der Spielsitzungen zu schützen (sichere Bereitstellung). Oder entscheiden Sie sich dafür, Flotteninstanzen unabhängig von der Aktivität der Spielsitzungen zu ersetzen (unsicherer Einsatz). Verwenden Sie während der Entwicklungs- und Testphasen unsichere Bereitstellungen, um die Bereitstellungszeit zu verkürzen.
 - Minimaler gesunder Prozentsatz. Geben Sie den Prozentsatz fehlerfreier Aufgaben an, die Sie während der Bereitstellung beibehalten möchten. Mit dieser Funktion können Sie entscheiden, wie viele Flotteninstanzen während einer Bereitstellung betroffen sind. Ein niedriger Wert priorisiert die Bereitstellungsgeschwindigkeit, während ein hoher Wert sicherstellt, dass die Verfügbarkeit der Spieleserver während der gesamten Bereitstellung hoch bleibt.
 - Strategie für Fehlschläge bei der Bereitstellung. Entscheiden Sie, welche Maßnahmen ergriffen werden sollen, wenn eine Bereitstellung fehlschlägt. Ein Bereitstellungsfehler bedeutet, dass einige der aktualisierten Container die Statusprüfungen nicht bestanden haben und daher als beeinträchtigt gelten. Sie können Bereitstellungen so einrichten, dass alle Flotteninstanzen automatisch auf den zuvor bereitgestellten Status zurückgesetzt werden. Alternativ können Sie sich dafür entscheiden, einige der beeinträchtigten Flotteninstanzen beizubehalten, um sie beim Debuggen zu verwenden.

Die Möglichkeit, aktive Flotten zu aktualisieren, ist äußerst nützlich, wenn du ein Update für deine Gameserver-Software bereitstellen möchtest. Nachdem du ein neues Container-Image für deinen Gameserver erstellt hast, besteht die Bereitstellung aus zwei Schritten: Aktualisiere zuerst die Container-Gruppendefinition mit dem neuen Image und zweitens aktualisiere die Container-Flotte. Amazon GameLift Server erledigt alle anderen Aufgaben nach Bedarf.

Verpackung von Behältern

Bei der Entwicklung Ihrer Containerstruktur für den Einsatz in einer Containerflotte besteht ein gemeinsames Ziel darin, die verfügbare Rechenleistung optimal zu nutzen. Um dieses Ziel zu erreichen, sollten Sie so viele Spielserver-Containergruppen wie möglich auf jede Flotteninstanz packen.

Amazon GameLift Servers hilft Ihnen dabei, indem es die maximale Anzahl an Gameserver-Containergruppen pro Instanz auf der Grundlage der folgenden Informationen berechnet:

- Der Instance-Typ der Flotte und ihre vCPU- und Speicherressourcen.
- Die vCPU- und Speicheranforderungen für alle Container in der Spielserver-Containergruppe.

Die vCPU- und Speicheranforderungen für alle Container in einer Containergruppe pro Instanz, falls es eine gibt.

Wenn Sie eine Containerflotte erstellen, können Sie das berechnete Maximum verwenden oder eine gewünschte Anzahl angeben. Es empfiehlt sich, mit Ihrer containerisierten Gameserver-Software zu experimentieren, um die Ressourcenanforderungen für eine optimale Gameserverleistung zu ermitteln.

Kapazitätsskalierung

Die Flottenkapazität gibt die Anzahl der Spielsitzungen an, die eine Flotte gleichzeitig veranstalten kann. Sie können die Kapazität auch anhand der Anzahl gleichzeitiger Spieler messen, die eine Flotte unterstützen kann. Um die Hosting-Kapazität einer Flotte zu erhöhen oder zu verringern, fügen Sie Flotteninstanzen hinzu oder entfernen sie.

Eine Containerflotte ist so konfiguriert, dass sie auf jeder Flotteninstanz eine bestimmte Anzahl gleichzeitiger Spielserverprozesse ausführt. (Sie können dies auf der Grundlage von (1) den Spielserver-Containergruppen pro Instanz und (2) der Anzahl der Spielserverprozesse, die in jeder Containergruppe ausgeführt werden, berechnen.) Die Anzahl der gleichzeitigen Spieleserver pro Instanz gibt Aufschluss darüber, welche Auswirkungen das Hinzufügen oder Entfernen der einzelnen Flotteninstanzen hat. Wenn Ihre Containerflotte beispielsweise einen Spieleserver-Prozess in jeder Spielserver-Containergruppe ausführt und jede Flotteninstanz 100 Gameserver-Containergruppen umfasst, erhöhen oder verringern Sie die Kapazität Ihrer Flotte für die Durchführung gleichzeitiger Spielsitzungen um jeweils 100. Wenn jede Spielsitzung über 10 Spielerplätze verfügt, erhöhen oder verringern Sie die Kapazität Ihrer Flotte, Spieler aufzunehmen, um jeweils 1000.

Bei Containerflotten können Sie jede der von bereitgestellten Methoden zur Kapazitätsskalierung verwenden. Amazon GameLift Servers Dazu zählen:

- Stellen Sie die Flottenkapazität manuell ein, indem Sie die gewünschte Anzahl an Flotteninstanzen festlegen.
- Richten Sie die automatische Skalierung ein, indem Sie auf einen gewünschten Puffer verfügbarer Instanzen abzielen (Zielverfolgung). Bei dieser Methode wird automatisch eine bestimmte Menge ungenutzter Hosting-Ressourcen beibehalten, sodass neue Spieler schnell in die Spiele einsteigen können. Wenn die Nachfrage der Spieler steigt oder sinkt, wird die Größe dieses Puffers kontinuierlich angepasst.
- Richten Sie die automatische Skalierung mit benutzerdefinierten Skalierungsregeln ein (erweiterte Funktion). Mit dieser Methode können Sie auf der Grundlage der von Ihnen ausgewählten Flottenkennzahlen skalieren.

client/server Verbindungen zum Spiel

Bei Amazon GameLift Servers verwalteten Flotten stellen Spieleclients eine direkte Verbindung zu deinen in der Cloud gehosteten Spielserversn her. Wenn ein Spielclient darum bittet, einem Spiel beizutreten, eine Spielsitzung Amazon GameLift Servers findet und Verbindungsinformationen (IP und Port) an den Spielclient weitergibt. Du kannst den externen Zugriff auf Flotteninstanzen kontrollieren, indem du bestimmte Portbereiche (eingehende Zugriffsberechtigungen) für die Flotte öffnest. Eingehende Berechtigungen bestimmen, welche Ports für eingehenden Datenverkehr geöffnet sind. Sie können schnell alle Ports herunterfahren, auf einige beschränken oder alle Ports öffnen.

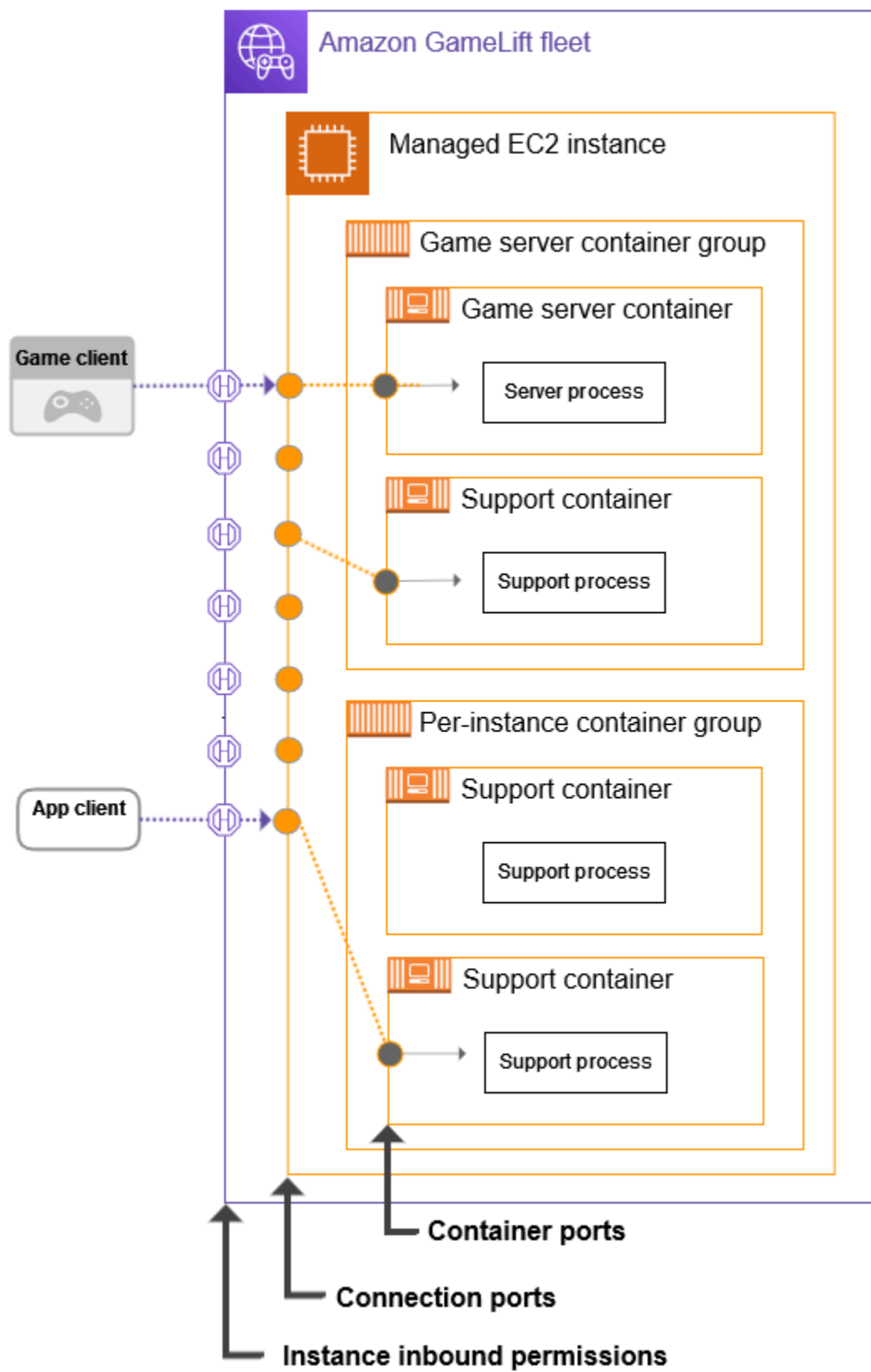
Verwaltete Containerflotten erfordern eine zusätzliche Einstellung, die den Zugriff auf Prozesse ermöglicht, die in einem Container ausgeführt werden. Wenn Sie eine Containerdefinition erstellen, geben Sie eine Reihe von Ports an, einen für jeden Prozess, der eine Verbindung aufnimmt. Dies umfasst:

- Alle Gameserver-Prozesse, die gleichzeitig im Gameserver-Container ausgeführt werden. Alle Spielserverprozesse müssen es Spieleclients ermöglichen, eine Verbindung herzustellen, um an einer Spielsitzung teilzunehmen.
- Jeder Prozess in einem Support-Container, zu dem eine externe Quelle eine Verbindung herstellen muss. Sie könnten beispielsweise eine Remoteverbindung zu einer Test-App herstellen.

Wenn du die Port-Einstellungen für den nach innen gerichteten Container festlegst, berechnet anhand dieser Einstellungen die nach außen gerichteten Eingangsberechtigungen, mit denen Spieleclients und andere Amazon GameLift Servers Anwendungen eine Verbindung herstellen können. Amazon GameLift Servers verwaltet außerdem die Zuordnung zwischen eingehenden Berechtigungen und einzelnen Container-Ports, sodass Spieler auf eine Spielsitzung in einem Container zugreifen können. Diese interne Zuordnung bietet eine Sicherheitsebene, indem sie deine Spieleserver vor direktem Zugriff auf die Container-Ports schützt. Du hast die Möglichkeit, die nach außen gerichteten Port-Einstellungen einer Flotte nach Bedarf anzupassen. Weitere Informationen zur manuellen Einstellung von Häfen für Containerflotten finden Sie unter. [Netzwerkverbindungen konfigurieren](#)

Sie können die Hafeneinstellungen einer Containerflotte jederzeit ändern. Für diese Änderung ist ein Flottenupdate erforderlich.

Das folgende Diagramm veranschaulicht die Rolle von Hafenverbindungen in einer Containerflotte. Wie in der Abbildung gezeigt, legen Sie Ports für einzelne Container fest und Amazon GameLift Servers verwenden diese Informationen, um genügend Ports auf der Flotteninstanz zu konfigurieren, um sie jedem Containerhafen zuzuordnen. Sowohl die eingehenden Zugriffsberechtigungen für externe Instances als auch die internen Verbindungsports werden Amazon GameLift Servers für Ihre Flotte berechnet, sofern Sie sie nicht manuell festlegen.



Protokollierung von Containern

In verwalteten Containerflotten werden die Standardausgabeströme (und Standardfehlerströme) für alle Container erfasst. Dazu gehören auch die Sitzungsprotokolle Ihres Spieleservers. Sie können eine Containerflotte so konfigurieren, dass sie eine von mehreren Optionen für die Verarbeitung von Ausgangsströmen verwendet:

- Speichern Sie die Container-Ausgabe als CloudWatch Amazon-Log-Stream. Jeder Logstream verweist auf die Flotten-ID und den Container. Wenn Sie diese Protokollierungsoption für die Flotte wählen, geben Sie eine CloudWatch Protokollgruppe an, in der alle Protokolldatenströme der Flotte organisiert werden. Anschließend können Sie CloudWatch Funktionen verwenden, um Protokolldaten nach Bedarf zu suchen und zu analysieren.
- Speichern Sie die Container-Ausgabe in einem Amazon Simple Storage Service (Amazon S3) - Speicher-Bucket. Sie können den Inhalt nach Bedarf ansehen, teilen oder herunterladen.
- Schalten Sie die Protokollierung aus. In diesem Szenario wird die Containerausgabe nicht gespeichert.

Amazon GameLift Servers sendet Protokolldaten von verwalteten Containerflotten an die CloudWatch oder Amazon S3 S3-Services in Ihrem AWS Konto. Um Ihre Daten einzusehen, verwenden Sie die AWS-Managementkonsole oder andere Tools, indem Sie sich bei Ihrem AWS Konto anmelden und mit den einzelnen Diensten arbeiten. Sie erweitern den eingeschränkten Amazon GameLift Servers Zugriff auf diese Aktionen, indem Sie eine Servicerolle für Ihre Containerflotten einrichten.

Sie können die Logging-Konfiguration einer Containerflotte jederzeit ändern. Für diese Änderung ist ein Flottenupdate erforderlich.

Containerflotten und der Agent Amazon GameLift Servers

Eine häufig verwendete Container-Architektur führt in jedem Container einen einzelnen Prozess aus. In einer Amazon GameLift Servers Containerflotte hat die Spielserver-Containergruppe einen Spieleserver-Container, der einen Spieleserver-Prozess ausführt. Amazon GameLift Servers verwaltet mit dieser Architektur den Lebenszyklus eines einzelnen Spieleserver-Prozesses in jeder Spielserver-Containergruppe auf einer Flotteninstanz.

Wenn Sie sich dafür entscheiden, eine Container-Architektur zu erstellen, die mehrere Spieleserver-Prozesse in jeder Spielserver-Containergruppe ausführt, benötigen Sie eine Möglichkeit, den Lebenszyklus aller Prozesse zu verwalten. Dazu gehören Aufgaben wie das Starten, Herunterfahren

und Ersetzen von Prozessen nach Bedarf, das Verwalten einer gewünschten Anzahl von Prozessen, die gleichzeitig ausgeführt werden sollen, und das Behandeln von Fehlerzuständen.

Sie können wählen, ob Sie den Amazon GameLift Servers Agenten für diese Aufgaben verwenden möchten. Für eine Containerflotte implementiert der Agent Laufzeitanweisungen, die angeben, welche ausführbaren Dateien ausgeführt werden sollen (und wie viele), Startparameter bereitstellen und Regeln für die Aktivierung von Spielservern festlegen. Zum Beispiel könnten Runtime-Anweisungen den Agenten anweisen, zehn Gameserverprozesse für den produktiven Einsatz und einen Gameserverprozess mit speziellen Startparametern für Tests aufrechtzuerhalten.

Um den Agenten mit Ihren Containerflotten zu verwenden, fügen Sie den Agenten Ihrem Container-Image hinzu und fügen Sie eine Reihe von Laufzeitanweisungen hinzu. Weitere Informationen über den Agenten finden Sie unter [Arbeite mit dem Amazon GameLift Servers Agenten](#).

Erstellen Sie eine Amazon GameLift Servers verwaltete Containerflotte

Erstellen Sie eine Amazon GameLift Servers verwaltete Containerflotte, um Ihren containerisierten Spieleserver in der AWS Cloud bereitzustellen und zu hosten. Wenn du eine Containerflotte erstellst, gib Containergruppendefinitionen an, die ein oder mehrere Container-Images (mindestens eines, das deinen Gameserver-Build enthält) und Konfigurationseinstellungen angeben.

Wenn Sie eine neue verwaltete Containerflottenressource erstellen, leiten Sie sofort die erste Phase der Flottenerstellung ein. Die verwaltete Flottenerstellung durchläuft mehrere Phasen, indem sie eine EC2 Instanz bereitstellt, eine Laufzeitumgebung installiert, Ihre Containergruppen auf der Instance bereitstellt und mit dem Starten von Spielserverprozessen beginnt. Amazon GameLift Servers Abhängig von der Laufzeitumgebung, die Ihr Spielserver-Build Amazon GameLift Servers benötigt, stellt es die neueste Version des Amazon Machine Image (AMI) zum Zeitpunkt der Flottenerstellung bereit (und alle future Instances in der Flotte werden dieselbe Version verwenden). Sie können den Status einer Flotte in der Konsole oder mithilfe von AWS Command Line Interface (AWS CLI) überwachen. Wenn eine Flotte bereit ist, Spielsitzungen abzuhalten, ändert sich der Status auf ACTIVE. Hilfe bei Problemen mit der Flottenerstellung finden Sie unter [Debuggen von Amazon GameLift Servers-Flottenproblemen](#).

Sie können sich dafür entscheiden, eine leere Containerflotte zu erstellen und die Containergruppendefinitionen der Flotte später hinzuzufügen oder zu aktualisieren. Wenn Sie eine Flotte ohne Containergruppendefinition erstellen, erreicht die Flotte nicht den aktiven Status.

 Note

Als bewährte Methode empfehlen wir, Ihre Flotten alle 30 Tage auszutauschen, um eine sichere up-to-date Laufzeitumgebung für Ihre gehosteten Spieleserver aufrechtzuerhalten. Dazu musst du eine neue Flotte erstellen und den Spielerverkehr dorthin migrieren. Weitere Anleitungen finden Sie unter [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#).

Verwenden Sie die [Amazon GameLift ServersKonsole](#) oder das Symbol AWS Command Line Interface (AWS CLI), um eine Containerflotte zu erstellen.

Console

Wählen Sie in der [Amazon GameLift ServersKonsole](#) den AWS-Region Ort aus, an dem Sie die Flotte erstellen möchten. Die Definitionen der Containergruppen müssen sich in derselben Region befinden, in der Sie die Flotte erstellen möchten.

Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie **Verwaltete Container: Flotten** aus. Wählen Sie auf der Seite Flotten die Option **Containerflotte erstellen** aus.

Schritt 1: Definieren Sie die Details der verwalteten Containerflotte

1. Geben Sie im Abschnitt Details zur Containerflotte eine Beschreibung der Flotte ein.
2. Geben Sie eine IAM-Rolle für die Flotte an. Diese Rolle verfügt über Berechtigungen, über die Sie verfügen Amazon GameLift Servers müssen, um die Containerflotte in Ihrem Namen zu verwalten. Hilfe beim Erstellen der erforderlichen Servicerolle finden Sie unter [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#).
3. Wählen Sie eine Protokollkonfigurationsoption. Die CloudWatch Option ist standardmäßig ausgewählt. Geben Sie die erforderlichen Informationen auf der Grundlage der ausgewählten Option ein.
4. Fügen Sie der Flotte Containergruppen hinzu. Dieser Schritt ist optional. Sie können sich dafür entscheiden, eine Flotte ohne Containergruppe zu erstellen und diese später hinzuzufügen. Eine Flotte ohne Containergruppen stellt keine Flotteninstanzen bereit und kann noch keine Spiele hosten, aber die Flottenressource wird erstellt.
 - Wählen Sie eine Spielserver-Container-Gruppendefinition aus. Geben Sie optional die Version der Definition an, die Sie bereitstellen möchten. Wenn Sie die Versionsnummer

nicht angeben, Amazon GameLift Servers wird automatisch die neueste Version verwendet.

- Fügen Sie optional eine Container-Gruppendefinition und -version pro Instanz hinzu. Wenn Sie die Versionsnummer nicht angeben, Amazon GameLift Servers wird automatisch die neueste Version verwendet.
5. In den zusätzlichen Details können Sie einige optionale Anpassungen vornehmen. Keine dieser Einstellungen ist erforderlich, um die Containerflotte zu erstellen.

Schritt 2: Definieren Sie Instanzdetails

1. Wählen Sie unter Instanzbereitstellung einen oder mehrere Remote-Standorte aus, an denen Instanzen bereitgestellt werden sollen. Die Heimatregion wird automatisch ausgewählt (dies ist die Region, in der Sie die Flotte erstellen). Wenn Sie zusätzliche Standorte auswählen, werden Flotteninstanzen auch an diesen Standorten bereitgestellt.

Important

Um Regionen zu verwenden, die nicht standardmäßig aktiviert sind, aktivieren Sie sie in Ihrem AWS-Konto.

- Flotten mit Regionen, die nicht aktiviert sind und die Sie vor dem 28. Februar 2022 erstellt haben, sind von dieser Anforderung nicht betroffen.
- Um neue Flotten mit mehreren Standorten zu erstellen oder bestehende Flotten mit mehreren Standorten zu aktualisieren, aktivieren Sie zunächst alle Regionen oder Local Zones, die Sie verwenden möchten.

[Weitere Informationen zu Regionen, die standardmäßig nicht aktiviert sind, und zu deren Aktivierung finden Sie unter Verwaltung in der. AWS-RegionenAllgemeine AWS-Referenz](#) Weitere Informationen finden Sie unter [Erste Schritte mit Local Zones](#) im AWS Local Zones Zones-Benutzerhandbuch.

2. Wählen Sie eine Instanzkonfiguration für die Flotte aus. Die Konsole berechnet automatisch die mindestens erforderliche vCPU und den erforderlichen Arbeitsspeicher (basierend auf den Gesamtlimits, die Sie für jede Containergruppe festlegen). Sie filtert die vollständige Liste der verfügbaren Instance-Typen auf der Grundlage der Ressourcenanforderungen und der von Ihnen eingegebenen Standorte. Sie können bei Bedarf weitere Filter hinzufügen.

Weitere Informationen zur Auswahl eines Instance-Typs finden Sie unter [Konfigurieren Sie eine Containerflotte](#). Die Größe des ausgewählten Instance-Typs wirkt sich darauf aus, wie Spielservers-Containergruppen auf jede Flotteninstanz gepackt werden. Je nach Ihrer Wahl sollten Sie Ihre Einstellung für die gewünschten Gameserver-Containergruppen pro Instance überprüfen.

Schritt 4: Überprüfen und Erstellen

- Überprüfe deine Flottenkonfigurationseinstellungen.

Sie können die Metadaten und die Konfiguration der Flotte jederzeit aktualisieren, unabhängig vom Flottenstatus. Weitere Informationen finden Sie unter [Eine Amazon GameLift Servers Flottenkonfiguration aktualisieren](#). Sie können die Flottenkapazität aktualisieren, nachdem die Flotte den Status AKTIV erreicht hat. Weitere Informationen finden Sie unter [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#). Sie können auch entfernte Standorte hinzufügen oder entfernen.

Wenn Sie mit der Überprüfung fertig sind, wählen Sie Erstellen.

Wenn Ihre Anfrage erfolgreich ist, zeigt die Konsole die Detailseite für die neue Flottenressource an. Anfänglich lautet der Status NEW, wie der Flottenerstellungsprozess Amazon GameLift Servers beginnt. Sie können den Status der neuen Flotte auf der Seite Fleets (Flotten) verfolgen. Eine Flotte ist bereit, Spielsitzungen abzuhalten, sobald sie den Status erreicht hat ACTIVE.

AWS CLI

Um eine Containerflotte mit dem zu erstellen AWS CLI, öffnen Sie ein Befehlszeilenfenster und verwenden Sie den `create-container-fleet` Befehl. Weitere Informationen zu diesem Befehl finden Sie [create-container-fleet](#) in der AWS CLI Befehlsreferenz.

Mit der unten abgebildeten `create-container-fleet` Beispielanforderung wird eine neue Containerflotte mit den folgenden Merkmalen erstellt:

- Das `ContainerGroupsConfiguration` gibt nur eine Spielservers-Container-Gruppengröße an: `MyAdventureGameContainerGroup`. Die Anzahl der Spielservers-Containergruppen, die für jede Flotteninstanz bereitgestellt werden, wird berechnet nach Amazon GameLift Servers.

- Die Flotte verwendet standardmäßig c5.large On-Demand-Instances.
- Standardmäßig öffnet die Flotte eine Reihe von Verbindungsports und Ports für eingehende Berechtigungen, berechnet von. Amazon GameLift Servers Sie stellt Containergruppen an den folgenden Orten bereit:

```
aws gamelift create-container-fleet \  
  --fleet-role-arn arn:aws:iam::MyAccount:role/MyContainersRole \  
  --game-server-container-group-definition-name "rn:aws:gamelift:us-  
west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2" \  

```

Wenn die Anfrage „Flotte erstellen“ erfolgreich ist, wird ein Satz von Flottenattributen Amazon GameLift Servers zurückgegeben, der die von Ihnen angeforderten Konfigurationseinstellungen und eine neue Containerflotten-ID enthält. Amazon GameLift Servers setzt dann den Flottenstatus und den Standortstatus auf Neu und leitet den Flottenaktivierungsprozess ein. Sie können den Status der Flotte nachverfolgen und andere Informationen zu der Flotte über die folgenden CLI-Befehle anzeigen:

- [describe-fleet-events](#)
- [describe-container-fleet](#)
- [describe-fleet-capacity](#)
- [describe-fleet-port-settings](#)
- [describe-fleet-utilization](#)
- [describe-fleet-location-capacity](#)
- [describe-fleet-location-utilization](#)

Mit diesen Befehlen können Sie die Kapazität der Flotte und andere Konfigurationseinstellungen nach Bedarf ändern:

- [update-container-fleet](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Erstellen Sie eine Containergruppendefinition für eine Amazon GameLift Servers Containerflotte

Eine Container-Gruppendefinition beschreibt, wie Sie Ihre containerisierten Gameserver-Anwendungen in einer Containerflotte bereitstellen. Es ist ein Blueprint, der festlegt, Amazon GameLift Servers welche Container-Images für die Flotte bereitgestellt werden sollen und wie sie ausgeführt werden. Wenn Sie eine Containerflotte erstellen, geben Sie die Containergruppendefinitionen an, die für die Flotte bereitgestellt werden sollen. Weitere Informationen zu Containergruppen finden Sie unter [Komponenten der Containerflotte](#).

Bevor Sie beginnen

Tipps dazu, was Sie tun sollten, bevor Sie mit der Erstellung einer Container-Gruppendefinition beginnen:

- Finalisieren Sie Ihre Container-Images und übertragen Sie sie in ein Amazon Elastic Container Registry (Amazon ECR) -Repository, in AWS-Region dem Sie die Container-Gruppe erstellen möchten. Amazon GameLift Servers erfasst zum Zeitpunkt der Erstellung der Container-Gruppendefinition einen Snapshot von jedem Image und verwendet den Snapshot bei der Bereitstellung in einer Containerflotte. Siehe [Erstellen Sie ein Container-Image für Amazon GameLift Servers](#).
- Erstellen Sie Ihre Containerdefinitionen als JSON-Dateien. Eine Container-Gruppendefinition umfasst eine oder mehrere Containerdefinitionen. Sie können die JSON-Dateien verwenden, wenn Sie eine Container-Gruppendefinition mit der erstellen AWS CLI für Amazon GameLift Servers.
- Stellen Sie sicher, dass Ihr AWS Benutzer über IAM-Berechtigungen für den Zugriff auf das Amazon ECR-Repository verfügt. Siehe [Beispiele für IAM-Berechtigungen für Amazon GameLift Servers](#).

Erstellen Sie eine Spielservers-Container-Gruppendefinition

Eine Spielservers-Containergruppe führt Ihre Spielservers-Software aus. Eine Spielservers-Containergruppe hat einen Gameserver-Container, auf dem die ausführbare Spielservers-Datei ausgeführt wird. Sie kann auch über einen oder mehrere Support-Container verfügen, in denen zusätzliche Software zur Unterstützung Ihres Spielservers ausgeführt werden kann. (Diese werden manchmal als „Sidecar“-Container bezeichnet.)

In diesem Thema wird beschrieben, wie Sie mithilfe der Amazon GameLift Servers Konsolen- oder AWS CLI-Tools eine einfache Gameserver-Container-Gruppendefinition erstellen. Ausführlichere

Informationen zu optionalen Funktionen finden Sie unter [Passen Sie eine Amazon GameLift Servers Containerflotte an](#).

 Note

Sie können die meisten Container-Gruppenseiten und Container-Definitionseinstellungen ändern, nachdem Sie sie erstellt haben. Wenn Sie Änderungen an einer Container-Definition vornehmen, wird Amazon GameLift Servers ein neuer Snapshot der aktualisierten Container-Images erstellt.

So erstellen Sie eine einfache Gameserver-Container-Gruppenseite:

Die folgenden Anweisungen beschreiben, wie Sie eine Container-Gruppenseite mit den minimal erforderlichen Parametern und unter Verwendung der Amazon GameLift Servers Standardwerte erstellen.

Console

Wählen Sie in der [Amazon GameLift Servers-Konsole](#) den AWS-Region Ort aus, an dem Sie die Containergruppe erstellen möchten.

Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie **Verwaltete Container-Gruppenseiten**. Wählen Sie auf der Definitionseite für Containergruppen die Option **Gruppenseite erstellen** aus.

Schritt 1: Definieren Sie die Details der Container-Gruppenseite

1. Geben Sie einen Definitionsnamen für eine Container-Gruppe ein. Der Name muss für die Region AWS-Konto und eindeutig sein.
2. Wählen Sie den Container-Gruppentyp **Spieleserver** aus.
3. Geben Sie unter **Gesamtspeicherlimit** die maximale Anzahl an Speicherressourcen ein, die für alle Container in der Containergruppe verfügbar sein sollen. Hilfe zur Berechnung dieses Werts finden Sie unter [Legen Sie Ressourcenlimits fest](#).
4. Geben Sie für **Total vCPU limit** die maximale Rechenleistung ein, die für alle Container in der Containergruppe verfügbar sein soll. Hilfe zur Berechnung dieses Werts finden Sie unter [Legen Sie Ressourcenlimits fest](#).

Schritt 2: Fügen Sie Containerdefinitionen hinzu

Eine Spielserver-Containergruppe hat mindestens einen Spielserver-Container. In der Konsole ist die erste Container-Definition, die Sie erstellen, der Gameserver-Container. In diesem Schritt wird beschrieben, wie Sie die erforderlichen Mindesteinstellungen für eine Gameserver-Container-Definition definieren.

1. Geben Sie einen Namen für die Container-Definition ein. Jeder für die Gruppe definierte Container muss einen eindeutigen Namenswert haben.
2. Link zu einem Container-Image mit deinem Gameserver-Build. Geben Sie den Amazon ECR-Image-URI für ein Container-Image in einem öffentlichen oder privaten Repository ein. Sie können eines der folgenden Formate verwenden:
 - Nur Bild-URI: `[AWS-Konto].dkr.ecr.[AWS-Region].amazonaws.com/[repository ID]`
 - Bild-URI + Digest: `[AWS-Konto].dkr.ecr.[AWS-Region].amazonaws.com/[repository ID]@[digest]`
 - Bild-URI + Tag: `[AWS-Konto].dkr.ecr.[AWS-Region].amazonaws.com/[repository ID]:[tag]`
3. Geben Sie die Amazon GameLift Servers Server-SDK-Version an, die der Spielserver-Build verwendet. Für eine Containerflotte muss dieser Wert 5.2.0 oder höher sein.
4. Stellen Sie unter Interner Container-Portbereich das Protokoll ein und definieren Sie einen Portbereich. Die Bereichsgröße muss größer sein als die Anzahl der gleichzeitigen Spielserverprozesse, die in diesem Container ausgeführt werden. Wenn der Spielserver-Container nur einen Serverprozess pro Container ausführt, benötigt dieser Portbereich nur wenige Ports. Weitere Details finden Sie unter [Netzwerkverbindungen konfigurieren](#).
5. Fügen Sie nach Bedarf weitere Container hinzu, um zusätzliche Support-Software auszuführen. Zusätzliche Container werden automatisch als Support-Container ausgewiesen. Eine Spielserver-Containergruppe kann nur einen Spielserver-Container und bis zu acht Support-Container haben. Geben Sie die folgenden mindestens erforderlichen Einstellungen an:
 - Name der Containerdefinition
 - URI des ECR-Bilds.
 - Interne Container-Ports (Geben Sie dies nur an, wenn der Container über Prozesse verfügt, die Netzwerkzugriff benötigen.)

Schritt 3: Abhängigkeiten konfigurieren

- Wenn Ihre Container-Gruppendefinition mehr als einen Container enthält, können Sie optional Abhängigkeiten zwischen den Containern festlegen. Weitere Informationen finden Sie unter [Legen Sie Container-Abhängigkeiten fest](#).

Schritt 3: Überprüfen und Erstellen

1. Überprüfen Sie alle Ihre Definitionseinstellungen für Containergruppen. Verwenden Sie Bearbeiten, um Änderungen an einem beliebigen Abschnitt vorzunehmen, einschließlich jeder Ihrer Container-Definitionen für die Gruppe.
2. Wenn Sie mit der Überprüfung fertig sind, wählen Sie Erstellen.

Wenn Ihre Anfrage erfolgreich ist, zeigt die Konsole die Detailseite für die neue Container-Gruppen-Definitionsressource an. Anfänglich lautet der Status `COPYING`, als Amazon GameLift Servers beginnt, Schnappschüsse aller Container-Images für die Gruppe zu erstellen. Wenn diese Phase abgeschlossen ist, ändert sich der Status der Container-Gruppendefinition auf `READY`. Eine Containergruppendefinition muss den `READY` Status haben, bevor Sie damit eine Containerflotte erstellen können.

AWS CLI

Wenn Sie die AWS CLI verwenden, um eine Container-Gruppendefinition zu erstellen, verwalten Sie Ihre Container-Definitionsconfigurationen in einer separaten JSON Datei. Sie können in Ihrem CLI-Befehl auf die Datei verweisen. Schemabeispiele finden Sie unter [Erstellen Sie eine JSON Container-Definitionsdatei](#).

Erstellen Sie eine Container-Gruppendefinition

Verwenden Sie den `create-container-group-definition` CLI-Befehl, um eine neue Container-Gruppendefinition zu erstellen. Weitere Informationen zu diesem Befehl finden Sie [create-container-group-definition](#) in der AWS CLI-Befehlsreferenz.

Dieses Beispiel veranschaulicht eine Anforderung für eine Spielserver-Container-Gruppendefinition. Es wird davon ausgegangen, dass Sie eine JSON-Datei mit den Containerdefinitionen für diese Gruppe erstellt haben.

```
aws gamelift create-container-group-definition \
```



```
--name MyAdventureGameContainerGroup \  
--operating-system AMAZON_LINUX_2023 \  
--container-group-type GAME_SERVER \  
--total-memory-limit-mebibytes 4096 \  
--total-vcpu-limit 1 \  
--game-server-container-definition file://MyAdventureGameContainers.json
```

Erstellen Sie eine **JSON** Container-Definitionsdatei

Wenn Sie eine Container-Gruppendefinition erstellen, definieren Sie auch die Container für die Gruppe. Eine Container-Definition spezifiziert das Amazon ECR-Repository, in dem das Container-Image gespeichert ist, sowie optionale Konfigurationen für Netzwerkpports, Grenzwerte für die CPU- und Speicherauslastung und andere Einstellungen. Wir empfehlen, eine einzige JSON Datei mit den Konfigurationen für alle Container in einer Containergruppe zu erstellen. Die Pflege einer Datei ist nützlich, um diese kritischen Konfigurationen zu speichern, gemeinsam zu nutzen und Versionen nachzuverfolgen. Wenn Sie die AWS CLI verwenden, um Ihre Container-Gruppendefinitionen zu erstellen, können Sie im Befehl auf die Datei verweisen.

Um eine Container-Definition zu erstellen

1. Erstellen und öffnen Sie eine neue .JSON Datei. Beispiel:

```
[~/work/glc]$ vim SimpleServer.json
```

2. Erstellen Sie für jeden Container der Gruppe eine separate Containerdefinition. Kopieren Sie den folgenden Beispielinhalt und ändern Sie ihn nach Bedarf für Ihre Container. Einzelheiten zur Syntax einer Container-Definition finden Sie [ContainerDefinitionInput](#) in der Amazon GameLift ServersAPI-Referenz.
3. Speichern Sie die Datei lokal, damit Sie in einem AWS CLI-Befehl darauf verweisen können.

Beispiel: Container-Definition für einen Spieleserver

Example

Dieses Beispiel beschreibt den wesentlichen Container für Ihre Gameserver-Containergruppe. Der Container „Essential Replica“ umfasst Ihre Gameserver-Anwendung, den Amazon GameLift Servers Agenten, und kann auch andere unterstützende Software für Ihr Spiele-Hosting enthalten. Die Definition muss einen Namen, eine Bild-URI und eine Portkonfiguration enthalten. In diesem Beispiel werden auch einige containerspezifische Ressourcenlimits festgelegt.

```
{
  "ContainerName": "MyAdventureGameServer",
  "ImageUri": "111122223333.dkr.ecr.us-east-1.amazonaws.com/gl-
containers:myadventuregame-server",
  "PortConfiguration": {
    "ContainerPortRanges": [
      {
        "FromPort": 2000,
        "Protocol": "TCP",
        "ToPort": 2010
      }
    ]
  },
  "ServerSdkVersion": "5.2.0"
}
```

Amazon GameLift ServersÜberall Flotten

Verwenden Sie Anywhere Flotten, wenn Sie Amazon GameLift Servers Funktionen mit Ihren eigenen Hosting-Ressourcen nutzen möchten. Anywhere-Flotten werden häufig als Testumgebungen für die iterative Entwicklung oder zusammen mit verwalteten Flotten in einer Hybrid-Hosting-Lösung verwendet.

Eine Anywhere-Flotte besteht aus einer Reihe von Rechenressourcen (virtuell oder physisch), die Sie bereitstellen und verwalten. Computer können sich an jedem beliebigen geografischen Standort mit Konnektivität befinden, von einem lokalen Laptop bis hin zu entfernten Außenposten. Wenn Sie eine Anywhere-Flotte einrichten, fügen Sie der Flotte Computer hinzu, indem Sie sie über registrieren. Amazon GameLift Servers Jeder Computer ist mit seiner IP-Adresse (oder seinem DNS-Namen) registriert, sodass eine Verbindung mit ihm hergestellt werden Amazon GameLift Servers kann.

Sie stellen Gameserver-Software für eine Anywhere-Flotte bereit, indem Sie sie auf jedem Computer installieren und Gameserver-Prozesse starten. Jeder gestartete Gameserver-Prozess stellt eine Verbindung zum Amazon GameLift Servers Dienst her und meldet die Bereitschaft, eine Spielsitzung abzuhalten. Sie können Ihre vorhandenen Tools für das Konfigurationsmanagement und die Bereitstellung verwenden, um die anfängliche Bereitstellung und die Hostverwaltung zu erledigen. Für die Verwendung mit sind einige zusätzliche Aufgaben erforderlichAmazon GameLift Servers, darunter:

- Computer registrieren und deregistrieren, um sie der Flotte hinzuzufügen oder aus ihr zu entfernen.

- Verwaltet up-to-date Authentifizierungstoken auf allen Computern. Serverprozesse auf dem Computer verwenden sie, wenn sie eine Verbindung zum Amazon GameLift Servers Dienst herstellen.

Note

Stellen Sie optional Ihre Anywhere-Flotte mit dem Amazon GameLift Servers Agenten bereit, um diese wichtigen Verwaltungsaufgaben zu automatisieren. Siehe [Arbeite mit dem Amazon GameLift Servers Agenten](#).

In den folgenden Themen erfahren Sie, wie Sie Anywhere-Flotten einrichten und verwalten können:

- [Entwicklungs-Roadmap für Hosting mit Anywhere Amazon GameLift Servers](#)
- [Entwicklungs-Roadmap für Hybrid-Hosting mit Amazon GameLift Servers](#)
- [Für iterative Entwicklung mit Amazon GameLift Servers Anywhere einrichten](#)
- [Erstelle eine Amazon GameLift Servers Überall Flotte](#)
- [Eine Amazon GameLift Servers Flottenkonfiguration aktualisieren](#)

Workflow zur Flottenerstellung an jedem Ort

Für Anywhere-Flotten Amazon GameLift Servers wird nur die Flottenressource eingerichtet. Sie richten Computer ein und registrieren sie bei der Flotte. Außerdem installieren Sie Gameserver-Software und starten Spielserversprozesse, um Spielesitzungen zu hosten.

1. Amazon GameLift Servers erstellt die Flottenressource in der Heimatregion der Flotte. Der Flottenstatus und der Status des benutzerdefinierten Standorts sind auf Neu gesetzt.
2. Amazon GameLift Servers beginnt mit dem Schreiben von Ereignissen in das Flottenereignisprotokoll.
3. Nachdem die Flottenressource erstellt wurde. Amazon GameLift Servers setzt den Flottenstatus auf Aktiv. Zu diesem Zeitpunkt können Sie neue Computer bei der Flotte registrieren.

Erstelle eine Amazon GameLift Servers Überall Flotte

In diesem Thema wird beschrieben, wie Sie ein erstellen Amazon GameLift Servers Überall Flotte. Mit einer Anywhere-Flotte können Sie Core verwenden Amazon GameLift Servers

Funktionen zur Verwaltung von Spielsitzungen beim Hosten von Spielsitzungen mit Ihren eigenen Rechenressourcen. Erstellen Sie eine Anywhere-Flotte für Ihre lokale Hardware oder andere cloudbasierte Ressourcen.

Anywhere-Flotten werden häufig zusammen verwendet Amazon GameLift Servers verwaltete Flotten in einer Hybrid-Hosting-Lösung. Sie bieten auch nützliche Testumgebungen bei der Entwicklung eines Spiels zum Hosten mit Amazon GameLift Servers. In diesen Themen erfahren Sie mehr darüber, wann und wie Sie eine Integration vornehmen sollten Amazon GameLift Servers Anywhere-Flotten in eine Game-Hosting-Lösung:

- [Hybrides Hosting](#)
- [Stellen Sie Hosting-Flotten bereit für Amazon GameLift Servers](#)
- [Für iterative Entwicklung mit Amazon GameLift Servers Anywhere einrichten](#)

Da Anywhere-Flotten selbst verwaltet werden, erfordert die Einrichtung einer Flotte einige zusätzliche Arbeit. Um eine Anywhere-Flotte für die Ausrichtung von Spielsitzungen und Spielern vorzubereiten, müssen Sie die folgenden Aufgaben erledigen:

Themen

- [Bevor Sie beginnen](#)
- [Erstellen Sie einen benutzerdefinierten Standort](#)
- [Erstellen Sie eine Anywhere-Flotte](#)
- [Fügen Sie der Flotte einen Computer hinzu](#)
- [Starte einen Spieleserver](#)

Bevor Sie beginnen

Bevor Sie eine Anywhere-Flotte erstellen, führen Sie die folgenden Aufgaben aus. Eine detailliertere Anleitung finden Sie unter [Entwicklungs-Roadmap für Hosting mit Anywhere Amazon GameLift Servers](#) oder [Entwicklungs-Roadmap für Hybrid-Hosting mit Amazon GameLift Servers](#).

- Integrieren Sie Ihren Gameserver-Code in den Amazon GameLift Servers Server-SDK Version 5.x (oder höher). Sie müssen nicht alle Aufgaben zur Spieleintegration erledigen, sondern nur die, die für einen Spieleserver-Build erforderlich sind. Es ist üblich, deinen lokalen Computer als Anywhere-Flotte einzurichten und deine Gameserver-Integration mithilfe einer Befehlszeilenschnittstelle zu testen (siehe [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#)). Sie können

zusätzliche Komponenten integrieren (z. B. Amazon GameLift Servers aktivierter Spieleclient), während du sie entwickelst.

- Package Sie Ihre Gameserver-Software für die Installation auf Ihren Anywhere-Flottencomputern. Das Paket sollte Ihren integrierten Gameserver-Build und die gesamte Support-Software enthalten, die Sie für den Betrieb Ihres Gameservers benötigen.
- Entscheiden Sie, ob Sie das verwenden möchten Amazon GameLift Servers Agent für Ihre Anywhere-Flotte. Der Agent ist ein computergestütztes Prozessmanagement-Tool, das einige der wichtigsten Aufgaben im Zusammenhang mit der Verwaltung von Serverprozessen und Berechnungen für die Verwendung mit automatisiert Amazon GameLift Servers. Weitere Informationen finden Sie unter [Arbeite mit dem Amazon GameLift Servers Agenten](#).

Erstellen Sie einen benutzerdefinierten Standort

Erstellen Sie einen benutzerdefinierten Standort, der den physischen Standort Ihrer Rechenressourcen darstellt. Wenn Sie eine Anywhere-Flotte erstellen, muss mindestens ein benutzerdefinierter Standort bereits definiert sein. Sie können jederzeit weitere benutzerdefinierte Standorte erstellen und diese zu einer vorhandenen Flotte hinzufügen.

Um einen benutzerdefinierten Standort zu erstellen

Verwenden Sie entweder Amazon GameLift Servers Konsole oder AWS Command Line Interface (AWS CLI), um einen benutzerdefinierten Speicherort zu erstellen.

Console

In der [Amazon GameLift Servers Konsole](#), verwenden Sie den Navigationsbereich, um die Seite Standorte zu öffnen. Wählen Sie „Standort erstellen“, um das Dialogfeld „Erstellen“ zu öffnen.

1. Geben Sie im Dialogfeld einen Standortnamen ein. Es hat sich bewährt, einen Namen zu verwenden, der einen aussagekräftigen Standort für eine Reihe von Rechenressourcen beschreibt. Dabei kann es sich um geografische Standorte, den Namen eines Rechenzentrums oder eine andere Standortkennung handeln. Amazon GameLift Servers hängt den Namen Ihres benutzerdefinierten Standorts mit custom- an.
2. (Optional) Fügen Sie Ihrem benutzerdefinierten Standort Tags hinzu. Jeder Tag (Markierung) besteht aus einem Schlüssel und einem optionalen Wert, beides können Sie bestimmen. Weisen Sie AWS Ressourcen, die Sie sinnvoll kategorisieren möchten, Tags zu, z. B. nach Zweck, Eigentümer oder Umgebung. Wählen Sie für jedes Tag, das Sie hinzufügen möchten, die Option Neues Tag hinzufügen aus.

3. Wählen Sie Create (Erstellen) aus.

AWS CLI

Erstellen Sie mit dem [create-location](#) Befehl eine benutzerdefinierte Position. Geben Sie einen `location-name` Wert an, der mit `beginnen muss` `custom-`. Es hat sich bewährt, einen Namen zu verwenden, der einen aussagekräftigen Standort für eine Reihe von Rechenressourcen beschreibt. Dabei kann es sich um geografische Standorte, den Namen eines Rechenzentrums oder eine andere Standortkennung handeln.

```
aws gamelift create-location \
  --location-name custom-location-1
```

Output

```
{
  "Location": {
    "LocationName": "custom-location-1",
    "LocationArn": "arn:aws:gamelift:us-east-1:111122223333:location/custom-
location-1"
  }
}
```

Erstellen Sie eine Anywhere-Flotte

Erstellen Sie eine Anywhere-Flotte für eine Reihe von Rechenressourcen, die Sie besitzen. Eine neue Anywhere-Flotte ist zunächst leer. Sie fügen der Flotte Rechenleistung hinzu, indem Sie sie registrieren.

Bei der Erstellung durchläuft eine neue Anywhere-Flotte schnell den Flottenstatus von `NEW` bis `ACTIVE`. Sie können der Flotte Recheneinheiten hinzufügen, wenn diese erreicht ist. `ACTIVE`

Um eine Anywhere-Flotte zu erstellen

Verwenden Sie entweder Amazon GameLift Servers Konsole oder AWS Command Line Interface (AWS CLI), um eine Anywhere-Flotte zu erstellen.

Console

In der [Amazon GameLift Servers Konsole](#), verwenden Sie den Navigationsbereich, um die Flotten-Seite zu öffnen. Wählen Sie Flotte erstellen, um den Workflow zur Flottenerstellung zu starten.

Schritt 1 Wählen Sie den Berechnungstyp

Wählen Sie die Option Anywhere und dann Weiter.

Schritt 2 Definieren Sie die Flottendetails

In diesem Schritt geben Sie einige wichtige Einstellungen für die gesamte Flotte an.

1. Füllen Sie den Abschnitt Flottendetails aus:
 - a. Geben Sie einen Flottenamen ein. Wir empfehlen, ein Flottenbenennungsmuster zu verwenden, das die Identifizierung von Flottentypen bei der Anzeige von Flottenlisten erleichtert.
 - b. Geben Sie eine kurze Beschreibung der Flotte an.
2. Geben Sie diese optionalen zusätzlichen Details nach Bedarf ein. Sie können diese Flotteneinstellungen später aktualisieren.
 - a. Wenn Sie eine Flotte für Produktionstests oder Tests vor der Produktion erstellen, verwenden Sie diese Einstellung, um einen Kostenwert pro Stunde für die Berechnungen der Flotte anzugeben. Amazon GameLift Servers kann diese Informationen während der Platzierung einer Spielsitzung verwenden, um Hosting-Ressourcen auf der Grundlage der Kosten auszuwählen.
 - b. Wenn Sie Metrikdaten für diese Flotte und andere kombinieren möchten, geben Sie einen Metrikgruppennamen an. Verwenden Sie für alle Flotten, die Sie kombinieren möchten, denselben Metrikgruppennamen. Zeigen Sie die Metriken für die Metrikgruppe an, um die aggregierten Daten zu sehen.
3. Fügen Sie optionale Tags zu Ihrem benutzerdefinierten Standort hinzu. Jeder Tag (Markierung) besteht aus einem Schlüssel und einem optionalen Wert, beides können Sie bestimmen. Weisen Sie AWS Ressourcen, die Sie sinnvoll kategorisieren möchten, Tags zu, z. B. nach Zweck, Eigentümer oder Umgebung. Wählen Sie für jedes Tag, das Sie hinzufügen möchten, die Option Neues Tag hinzufügen aus.
4. Wählen Sie Weiter, um den Workflow fortzusetzen.

Schritt 3: Wählen Sie benutzerdefinierte Speicherorte aus

Identifizieren Sie in diesem Schritt den physischen Standort der Computer, die Sie zu dieser Flotte hinzufügen möchten. Sie können jetzt einen oder mehrere Standorte angeben und später nach Bedarf Standorte hinzufügen oder entfernen.

1. Wählen Sie unter Benutzerdefinierte Standorte einen oder mehrere Standorte für die Berechnungen der Flotte aus. Die Liste enthält alle benutzerdefinierten Standorte, die an Ihren aktuell ausgewählten AWS-Region Standorten definiert wurden. Um einen neuen benutzerdefinierten Standort zu definieren, den Sie der Flotte hinzufügen möchten, wählen Sie Standort erstellen.
2. Wählen Sie Weiter, um den Workflow fortzusetzen.

Schritt 4 Überprüfen und erstellen

Überprüfen Sie Ihre Einstellungen, bevor Sie die Flotte erstellen.

Wenn Sie bereit sind, die neue Flotte einzusetzen, wählen Sie Create. Amazon GameLift Servers startet sofort den Flottenaktivierungsprozess, weist eine eindeutige ID zu und versetzt die Flotte in den NEW Status. Sie können den Fortschritt der Flotte auf der Seite Flotten verfolgen.

AWS CLI

Verwenden Sie den [create-fleet](#) Befehl, um eine Flotte vom Typ ANYWHERE Compute zu erstellen. Geben Sie einen Namen und mindestens einen benutzerdefinierten Speicherort an. Amazon GameLift Servers erstellt die Anywhere-Flottenressource in Ihrem aktuellen Standard AWS-Region (oder Sie können ein `--region`-Tag hinzufügen, um einen anderen anzugeben AWS-Region).

Die folgende Beispielanforderung erstellt eine neue Flotte mit den minimal erforderlichen Einstellungen. Ersetzen Sie *FleetName* und *custom-location* durch Ihre eigenen Informationen.

```
aws gamelift create-fleet \  
--name FleetName \  
--compute-type ANYWHERE \  
--locations "Location=custom-location"
```


Beispielantwort

```
{
  "FleetAttributes": {
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetArn": "arn:aws:gamelift:us-west-2:111122223333:fleet/fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "Name": "HardwareAnywhere",
    "CreationTime": "2023-02-23T17:57:42.293000+00:00",
    "Status": "ACTIVE",
    "MetricGroups": [
      "default"
    ],
    "CertificateConfiguration": {
      "CertificateType": "DISABLED"
    },
    "ComputeType": "ANYWHERE"
  }
}
```

Bei der Erstellung wird eine neue Anywhere-Flotte schnell in den Flottenstatus versetzt **ACTIVE**. Sie können der Flotte Computer hinzufügen, nachdem sie den Wert erreicht hat **ACTIVE**.

Beachten Sie, dass die Antwort die Standorte der Flotte nicht beinhaltet. Sie können die vollständigen Flottendetails abrufen, indem Sie [describe-fleet-attributes](#) und aufrufen [describe-fleet-location-attributes](#).

Fügen Sie der Flotte einen Computer hinzu

Um einer Flotte eine Rechenressource hinzuzufügen und sie für die Ausrichtung von Spielsitzungen vorzubereiten, führen Sie die folgenden Aufgaben aus:

- Registrieren Sie die Rechenleistung bei der Flotte. Die Registrierung sagt Amazon GameLift Servers welche physischen Hosting-Ressourcen Teil der Flotte sind.
- Fordern Sie ein Authentifizierungstoken für die Berechnung an. Jeder Spieleserver, der auf dem Computer läuft, benötigt dieses Token, um sich mit dem zu verbinden Amazon GameLift Servers Dienst. Authentifizierungstoken sind temporär und müssen regelmäßig aktualisiert werden.

 Note

Wenn du deine Gameserver-Software mit dem Amazon GameLift Servers Agent, Sie können diesen Schritt überspringen. Der Agent registriert automatisch jede Berechnung und verwaltet ein gültiges Authentifizierungstoken für die Berechnung. Siehe [Arbeite mit dem Amazon GameLift Servers Agenten](#).

Sie können eine Berechnung registrieren und ein Authentifizierungstoken anfordern, indem Sie die AWS CLI verwenden oder programmgesteuerte Aufrufe an das AWS SDK tätigen für Amazon GameLift Servers. Diese Aktionen sind nicht verfügbar über Amazon GameLift Servers console.

Als bewährte Methode empfehlen wir, diese beiden Aufgaben zu automatisieren, indem jedem Computer ein Startskript hinzugefügt wird. Das Startskript ruft automatisch sowohl die `get-compute-auth-token` Befehle als `register-compute` auch auf. Sie können auch Aufgaben automatisieren, um das Authentifizierungstoken während der gesamten Lebensdauer des Computers regelmäßig zu aktualisieren und den Computer beim Herunterfahren abzumelden.

Jede der Startaktionen gibt rechenspezifische Werte zurück, die Sie auf dem Computer speichern müssen. Wenn ein Spieleserverprozess auf dem Computer gestartet wird, muss er diese Werte als Serverparameter übergeben, wenn er eine Verbindung mit dem Amazon GameLift Servers Dienst (siehe [ServerParameters](#) in der Server-SDK-Referenz). Es wird empfohlen, diese rechenspezifischen Werte (oder ihre Speicherorte) als Umgebungsvariablen festzulegen. Wenn Sie das verwenden Amazon GameLift Servers Agent, diese Aufgabe wird für Sie erledigt. Die berechnungsspezifischen Werte lauten wie folgt:

- `register-compute` gibt einen Wert für zurück. `GameLiftServiceSdkEndpoint` Setzen Sie diesen Wert auf den `webSocketUrl` Serverparameter.
- `compute-auth-token` gibt das Authentifizierungstoken zurück. Setzen Sie diesen Wert auf den `authToken` Serverparameter.

AWS CLI

In den folgenden Anweisungen wird beschrieben, wie jede Anfrage mithilfe der AWS CLI manuell eingereicht wird.

Um einen Computer zu registrieren

Rufen Sie [register-compute](#) an, um einen Computer zu registrieren. Identifizieren Sie die ID der Flotte, zu der die Rechenleistung hinzugefügt werden soll. Geben Sie die folgenden Recheninformationen an: einen aussagekräftigen Namen, eine IP-Adresse und einen Standort. Der Standort des Computers muss ein benutzerdefinierter Standort sein, der bereits mit der Flotte verknüpft ist. Wenn Sie einen anderen benutzerdefinierten Standort verwenden möchten, verwenden Sie Amazon GameLift Servers Konsole, um die Flotte zu aktualisieren, oder rufen Sie den AWS CLI-Befehl [create-fleet-locations](#) auf, um der Flotte einen benutzerdefinierten Standort hinzuzufügen.

Ersetzen Sie im folgenden Beispiel die Platzhalterwerte für Ihre Rechenleistung und Flotte. Der `fleet-id` Wert wird zurückgegeben, wenn Sie eine Anywhere-Flotte erstellen. Sie können die vollständigen Flottendetails abrufen, indem Sie [describe-fleet-attributes](#) und anrufen [describe-fleet-location-attributes](#).

```
aws gamelift register-compute \  
  --compute-name HardwareAnywhere \  
  --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --ip-address 10.1.2.3 \  
  --location custom-location-1
```

Beispielausgabe

```
{  
  "Compute": {  
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "FleetArn": "arn:aws:gamelift:us-west-2:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "ComputeName": "HardwareAnywhere",  
    "ComputeArn": "arn:aws:gamelift:us-west-2:111122223333:compute/  
HardwareAnywhere",  
    "IpAddress": "10.1.2.3",  
    "ComputeStatus": "Active",  
    "Location": "custom-location-1",  
    "CreationTime": "2023-02-23T18:09:26.727000+00:00",  
    "GameLiftServiceSdkEndpoint": "wss://us-west-2.api.amazongamelift.com"  
  }  
}
```

Um ein Authentifizierungstoken anzufordern

Rufen Sie an [get-compute-auth-token](#), um ein gültiges Authentifizierungstoken anzufordern. Registrieren Sie einen Computer. Identifizieren Sie die Flotten-ID und den Computernamen.

Ersetzen Sie im folgenden Beispiel die Platzhalterwerte für Ihre Rechenleistung und Ihren Fuhrpark. Der `fleet-id` Wert wird zurückgegeben, wenn Sie eine Anywhere-Flotte erstellen. Sie können die vollständigen Flottendetails telefonisch abrufen [describe-fleet-attributes](#). Um Recheninformationen zu erhalten, rufen Sie [list-compute](#) mit der Flotten-ID an, um alle Computer zu sehen, die für die Flotte registriert sind.

```
aws gamelift get-compute-auth-token \  
  --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --compute-name HardwareAnywhere
```

Beispielausgabe

```
{  
  "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
  "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
  "ComputeName": "HardwareAnywhere",  
  "ComputeArn": "arn:aws:gamelift:us-east-1:111122223333:compute/  
HardwareAnywhere",  
  "AuthToken": "0c728041-3e84-4aaa-b927-a0fb202684c0",  
  "ExpirationTimestamp": "2023-02-23T18:47:54+00:00"  
}
```

Starte einen Spieleserver

Nachdem du eine Anywhere-Flotte erstellt und der Flotte einen oder mehrere Computer hinzugefügt hast, kannst du mit dem Betrieb deiner Spieleserver beginnen.

Schritt 1: Installiere deine Gameserver-Software

Installieren Sie Ihren Gameserver-Build und die gesamte abhängige Software auf jedem Computer in Ihrer Anywhere-Flotte. Der Gameserver-Build muss integriert sein Amazon GameLift Servers Server-SDK Version 5.x (oder höher) mit den mindestens erforderlichen Funktionen für die Kommunikation mit Amazon GameLift Servers Dienst.

Schritt 2 Bereite deine Computer für den Betrieb eines Spieleservers vor

Stellen Sie sicher, dass jeder Computer registriert ist und über ein gültiges Authentifizierungstoken verfügt. Wenn du Skripte zur Verwaltung dieser Aufgaben verwendest, stelle sicher, dass die Skripte auf jedem Computer ausgeführt werden, bevor du irgendwelche Spielserverprozesse startest.

Wenn du das bereitgestellt hast Amazon GameLift Servers Stellen Sie sicher, dass der Agent mit Ihrer Gameserver-Software gestartet wird.

Schritt 3 Starte einen Gameserver-Prozess

Führen Sie eine ausführbare Instanz Ihres Spieleservers auf einem Computer aus. Wenn Ihr Spieleserver-Build ordnungsgemäß integriert ist, ruft der Gameserver-Prozess die Server-SDK-Aktion `InitSDK()` mit einer Reihe gültiger Serverparameter auf. Wenn der Serverprozess bereit ist, eine Spielsitzung zu hosten, ruft er `onProcessReady()` auf.

Note

Wenn du deine Gameserver-Software mit dem bereitgestellt hast Amazon GameLift Servers Agent, Sie können diesen Schritt überspringen. Der Agent startet automatisch Gameserver-Prozesse auf der Grundlage der von Ihnen angegebenen Runtime-Anweisungen.

Sie können den Fortschritt überwachen, indem Sie sich die Serverprozess-Metriken für aktivierende und aktive Serverprozesse ansehen. Siehe [Amazon GameLift Servers-Metriken für Flotten](#). Wenn dein Spieleserverprozess nicht initialisiert werden kann, vergewissere dich, dass der Prozess die richtigen Serverparameterwerte für den Computer abrufen, auf dem er läuft.

Erstellen Sie eine Hybrid-Hosting-Lösung

Eine Hybrid-Hosting-Lösung kombiniert mehrere Quellen für Game-Hosting-Ressourcen, um dein Spiel zu hosten, einschließlich Amazon GameLift Servers verwalteter Flotten, die im Spiel laufen, AWS Cloud und Ressourcen, die du selbst bereitstellst und verwaltest. In diesem Thema werden einige gängige Muster für den Aufbau einer Hybridlösung beschrieben. Außerdem finden Sie Tipps, wie Sie selbstverwaltetes Spielehosting erfolgreich mit cloudbasiertem Game-Hosting kombinieren können, das von verwaltet wird. Amazon GameLift Servers

Gängige Hybridmuster

- **Kostenoptimierung:** Verwenden Sie für jedes Szenario den kostengünstigsten Flottentyp, z. B. Anywhere-Flotten für die Basiskapazität und verwaltete Flotten für Spitzennachfrage.
- **Geografische Flexibilität:** Stellen Sie verwaltete Flotten in stark frequentierten AWS-Regionen und überall verfügbaren Flotten bereit, in denen Sie über eine bestehende Infrastruktur oder spezifische Compliance-Anforderungen verfügen.
- **Risikominderung:** Reduzieren Sie die Abhängigkeit von einem einzelnen Hosting-Ansatz, indem Sie die Last auf mehrere Flottenarten und Anbieter verteilen.
- **Schrittweise Migration:** Umstellung von lokalen oder anderen Hosting-Lösungen auf AWS schrittweise unter Beibehaltung der Servicekontinuität.

Entwicklung und Produktion sind aufgeteilt

Verwenden Sie verschiedene Flottenarten für Entwicklungs- und Produktionsumgebungen:

- **Entwicklung:** Überall verfügbare Flotten für kostengünstige Entwicklung und Erprobung
- **Produktion:** Verwaltete Amazon EC2 - oder Containerflotten für skalierbares, zuverlässiges Produktionshosting

Regionale Optimierung

Optimieren Sie die Flottenarten anhand von AWS-Regionen Merkmalen:

- **Stark frequentiert AWS-Regionen:** Verwaltete Flotten mit auto-scaling für variable Nachfrage
- **Spezialisiert AWS-Regionen:** Überall verfügbare Flotten aus Gründen der Einhaltung von Vorschriften, Datenhoheit oder vorhandener Infrastruktur

Staffelung der Kapazität

Verwenden Sie unterschiedliche Flottenarten für unterschiedliche Kapazitätsstufen:

- **Basiskapazität:** Überall verfügbare Flotten oder Reserved Instances für vorhersehbare Auslastung
- **Burst-Kapazität:** Verwaltete Flotten mit auto-scaling für Spitzennachfrage
- **Überlaufkapazität:** Spot-Instances oder zusätzliche AWS-Regionen Instances für extreme Spitzenzeiten

Überlegungen zur Implementierung

Berücksichtigen Sie beim Aufbau einer Hybridlösung die folgenden Schlüsselfaktoren:

Konfiguration der Warteschlange für Spielsitzungen

Konfiguriere deine Warteschlangen für Spielsitzungen so, dass sie alle Flottenarten umfassen, und lege entsprechende Prioritäten und Latenzeinstellungen fest, um eine optimale Platzierung in deiner hybriden Infrastruktur zu gewährleisten.

Überwachung und Beobachtbarkeit

Implementieren Sie eine umfassende Überwachung aller Flottenarten, um den Überblick über Leistung, Kapazität und Kosten Ihrer Hybridlösung zu behalten.

Operative Komplexität

Berücksichtigen Sie die zunehmende betriebliche Komplexität bei der Verwaltung mehrerer Flottenarten, einschließlich unterschiedlicher Einsatzprozesse, Überwachungstools und Verfahren zur Fehlerbehebung.

Netzwerkkonnektivität

Sorgen Sie für eine zuverlässige Netzwerkkonnektivität zwischen Ihren verschiedenen Hosting-Umgebungen, insbesondere für Anywhere-Flotten, die sich vor Ort oder bei verschiedenen Cloud-Anbietern befinden können.

Erste Schritte mit Hybrid-Hosting

Um eine Hybrid-Hosting-Lösung zu implementieren:

1. Fangen Sie einfach an: Beginnen Sie mit einem einzigen Flottentyp und fügen Sie nach und nach weitere hinzu, wenn Ihre Anforderungen klarer werden.
2. Planen Sie Ihre Architektur: Entwerfen Sie Ihre Hybridarchitektur auf der Grundlage Ihrer spezifischen Anforderungen in Bezug auf Kosten, Leistung, Compliance und betriebliche Komplexität.
3. Warteschlangen konfigurieren: Richten Sie Warteschlangen für Spielsitzungen ein, die sich über Ihre verschiedenen Flottenarten erstrecken, mit entsprechenden Prioritäten und Platzierungsstrategien.
4. Gründlich testen: Testen Sie die Platzierung von Spielsitzungen und Failover-Szenarien in Ihrer gesamten hybriden Infrastruktur, bevor Sie mit der Produktion beginnen.

5. Überwachen und optimieren: Überwachen Sie kontinuierlich Leistung und Kosten für alle Flottenarten und passen Sie Ihre Konfiguration nach Bedarf an.

Tipps

- Verwende dieselben Spielclient- und Serverkomponenten für verwaltete und selbstverwaltete Hosting-Ressourcen. Sorgen Sie für ein einheitliches Spielerlebnis auf allen Hosting-Ressourcen.
- Verwenden Sie dieselben FlexMatch Matchmaker, um Spiele auf allen Hosting-Ressourcen zu platzieren.
- Verwalten Sie Ihre Hybrid-Hosting-Ressourcen zentral gemeinsam, während Sie sie auf der ganzen Welt einsetzen.
- Da die Nachfrage der Spieler schwankt, können Sie die Auslastung von Spielsitzungen nahtlos über verwaltete und selbstverwaltete Ressourcen hinweg verwalten.
- Mit dem Amazon GameLift Servers Agenten können Sie dieselben Tools verwenden, um die Lebenszyklen von Spieleservern auf allen Arten von Hosting-Ressourcen zu verwalten.
- Erfassen Sie Spiel- und Spielermetriken und Logs für alle Hosting-Ressourcen. Nutzen Sie Amazon GameLift Servers Funktionen und andere AWS Dienste, um Daten zu kombinieren und kohärente Observability-Lösungen zu entwickeln.

Platzierung der Spielsitzungen konfigurieren

Bei der Platzierung von Spielsitzungen werden verfügbare Spieleserver für die Ausrichtung neuer Spielsitzungen ausfindig gemacht. Amazon GameLift Servers verwendet Warteschlangen für Spielsitzungen, um Spielsitzungen intelligent über deine Flotten zu verteilen und dabei Faktoren wie Spielerlatenz, Flottenkapazität und Kostenoptimierung zu berücksichtigen.

Eine gut konfigurierte Warteschlange für Spielsitzungen stellt sicher, dass die Spieler mit dem besten verfügbaren Spieleserver verbunden sind, was für optimale Leistung sorgt und gleichzeitig Ihre Hosting-Ressourcen effizient nutzt. Warteschlangen können sich über mehrere Flotten und AWS Regionen erstrecken, um eine globale Abdeckung und Redundanz zu gewährleisten.

Eine Warteschlange für Spielsitzungen ist der wichtigste Mechanismus, Amazon GameLift Servers mit dem nach verfügbaren Spieleservern gesucht und diese als Gastgeber für neue Spielsitzungen ausgewählt werden. Warteschlangen bieten eine weitaus effizientere Möglichkeit, eine große Anzahl von Anfragen für Spielsitzungen zu verarbeiten und Platzierungen für sie in mehreren Flotten von

Hosting-Ressourcen zu finden. Wenn deine Hosting-Lösung mehr als eine Flotte verwendet und du große Mengen an Anfragen verarbeitest, benötigst du wahrscheinlich eine Warteschlange.

Wenn dein Spiel eine neue Spielsitzung für Spieler starten möchte, sendet es eine Platzierungsanfrage an den Amazon GameLift Servers Dienst, der sie in die Warteschlange weiterleitet. Die Konfiguration der Warteschlange bestimmt, wann und wie die Anfragen bearbeitet werden. Amazon GameLift Servers durchsucht bei der Bearbeitung einer Platzierungsanfrage eine Reihe von Flotten nach einem Spielservers, auf dem die Spielsitzung gehostet werden kann. Die Platzierung ist erfolgreich, wenn Amazon GameLift Servers ein verfügbarer Spielservers gefunden und dieser aufgefordert wird, eine Spielsitzung zu starten.

Themen

- [Eigenschaften der Warteschlange](#)
- [Bewährte Methoden für Warteschlangen bei Amazon GameLift Servers Spielsitzungen](#)
- [Eine Warteschlange für Spielsitzungen erstellen](#)
- [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#)

Eigenschaften der Warteschlange

Eine Warteschlange für Amazon GameLift Servers Spielsitzungen ist eine AWS Cloud-Ressource. Sie können in jeder Warteschlange AWS-Region, die Amazon GameLift Servers unterstützt, eine Warteschlange erstellen (siehe [Amazon GameLift Servers Servicestandorte](#)). Anfragen zur Platzierung von Spielsitzungen werden an diesen Standort gesendet und dort bearbeitet.

Die automatische Platzierung von Spielsitzungen mit Warteschlangen bietet sowohl für Spieleentwickler als auch für Spieler erhebliche Vorteile. Dazu zählen:

- Warteschlangen sorgen für die „bestmögliche“ Platzierung. Bei der Verarbeitung von Platzierungsanfragen für Spielsitzungen verwendet eine Warteschlange den Amazon GameLift Servers FleetIQ-Algorithmus, um Platzierungen auf der Grundlage definierter Einstellungen zu priorisieren, darunter Kosten, Standort und Spielerlatenz.
- Warteschlangen unterstützen Spot-Flotten, um die Kosten für das Hosting von Spielen zu senken. Du kannst deine Warteschlangen mit AWS Spot-Flotten, die oft deutlich niedrigere Hosting-Kosten bieten, sowie mit On-Demand-Flotten konfigurieren. Da niedrige Kosten eines der wichtigsten Kriterien für Platzierungen sind, können Warteschlangen immer Kostenunterschiede ausnutzen.
- In Warteschlangen können neue Spiele bei hoher Nachfrage schneller platziert werden. Indem du eine Warteschlange mit mehreren Flotten konfigurierst, bietest du flexiblere Optionen für die

Platzierung von Spielsitzungen. Zusätzliche Flotten bieten aber auch Backup-Kapazitäten, die bei steigendem Bedarf benötigt werden. Wenn bei jeder Platzierungsanfrage eine Spielsitzung nicht am bevorzugten Ort stattfinden Amazon GameLift Servers kann, wird automatisch mit der Bewertung anderer Standorte fortgefahren.

- Warteschlangen können die Verfügbarkeit von Spielservern belastbarer machen. Ausfälle können vorkommen. Bei einer Warteschlange mit mehreren Flotten muss eine Verlangsamung oder ein Ausfall den Spielerzugriff auf dein Spiel nicht beeinträchtigen. Indem du deine Warteschlange mit Flotten konfigurierst, die über Kapazitäten in verschiedenen Zonen AWS-Regionen und Verfügbarkeitszonen verfügen, kannst du sicherstellen, dass Spieler immer eine Spielsitzung finden, an der sie teilnehmen können.
- Erhalte Statistiken über die Platzierungen von Spielsitzungen und die Leistung in der Warteschlange. Amazon GameLift Servers gibt Queue-Metriken aus, darunter Statistiken über erfolgreiche und fehlgeschlagene Platzierungen, die Anzahl der Anfragen in der Warteschlange und die durchschnittliche Zeit, die Anfragen in der Warteschlange verbringen. Sie können diese Metriken in der Amazon GameLift Servers-Konsole oder in CloudWatch anzeigen.

Informationen zum Erstellen einer einfachen Startwarteschlange finden Sie unter [Eine Warteschlange für Spielsitzungen erstellen](#).

Themen

- [Bewährte Methoden für Warteschlangen bei Amazon GameLift Servers Spielsitzungen](#)
- [Eine Warteschlange für Spielsitzungen erstellen](#)
- [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#)

Bewährte Methoden für Warteschlangen bei Amazon GameLift Servers Spielsitzungen

Eine Warteschlange für Spielsitzungen enthält eine Liste von Flotten, in denen neue Spielsitzungen platziert Amazon GameLift Servers werden können. Jede Flotte kann über Hosting-Ressourcen verfügen, die an mehreren geografischen Standorten eingesetzt werden. Bei der Auswahl einer Platzierung wählt die Warteschlange eine Flotte und einen Flottenstandort auf der Grundlage einer Reihe von Prioritäten aus, die Sie für die Flotte festgelegt haben.

Beachten Sie die folgenden Richtlinien und bewährten Methoden:

- Füge Flotten an Orten hinzu, die deine Spieler abdecken. Du kannst Flotten und Aliase an jedem verfügbaren Ort hinzufügen. Der Standort ist wichtig, wenn du Platzierungen auf der Grundlage der gemeldeten Spielerlatenz vornimmst.
- Verwende Aliase für alle Flotten. Weisen Sie jeder Flotte in einer Warteschlange einen Alias zu und verwenden Sie die Aliasnamen, wenn Sie Ziele in Ihrer Warteschlange festlegen.
- Verwende für alle Flotten denselben oder einen ähnlichen Spiel-Build oder dasselbe Skript. In der Warteschlange können Spieler auf jeder beliebigen Flotte in der Warteschlange an Spielsitzungen teilnehmen. Die Spieler müssen in der Lage sein, an jeder Spielsitzung auf jeder Flotte teilzunehmen.
- Erstelle Flotten an mindestens zwei Orten. Indem Sie Spieleserver an mindestens einem anderen Standort hosten lassen, verringern Sie die Auswirkungen regionaler Ausfälle auf Ihre Spieler. Sie können Ihre Backup-Flotten herunterskalieren und Auto Scaling verwenden, um die Kapazität zu erhöhen, wenn die Nutzung zunimmt.
- Priorisieren Sie die Platzierung Ihrer Spielsitzungen. In einer Warteschlange werden Platzierungsoptionen anhand mehrerer Elemente priorisiert, einschließlich der Reihenfolge der Ziellisten.
- Erstellen Sie Ihre Warteschlange am selben Ort wie Ihr Kundenservice. Indem Sie Ihre Warteschlange an einem Ort in der Nähe Ihres Kundendienstes platzieren, können Sie die Kommunikationslatenz minimieren.
- Verwenden Sie Flotten mit mehreren Standorten. Verwenden Sie die Konfiguration des Warteschlangenfilters, um zu verhindern, dass die Warteschlange Spielsitzungen an bestimmten Orten platziert. Du kannst mindestens zwei Flotten mit mehreren Standorten und unterschiedlichen Heimatstandorten einsetzen, um die Auswirkungen von Spielplatzierungen während eines regionalen Ausfalls zu minimieren.
- Verwenden Sie dieselbe TLS-Zertifikatseinstellung für alle Flotten. Spieleclients, die eine Verbindung zu Spielsitzungen in Ihren Flotten herstellen, müssen über kompatible Kommunikationsprotokolle verfügen.

Eine Warteschlange für Spielsitzungen erstellen

Warteschlangen werden verwendet, um neue Spielsitzungen an mehreren Flotten und Standorten zu platzieren. Ihr Spiel startet neue Spielsitzungen, indem Sie Platzierungsanfragen an eine Warteschlange senden. Eine Warteschlange ist mit Anweisungen zur Bearbeitung von Anfragen konfiguriert. Weitere Informationen zum Initiieren von Platzierungsanfragen für Spielsitzungen finden Sie unter [Spielsitzungen erstellen](#).

Um eine Warteschlange für Spielsitzungen zu erstellen

Diese Anleitung veranschaulicht, wie Sie eine einfache Arbeitswarteschlange mit minimalen Konfigurationseinstellungen und Standardeinstellungen erstellen. Es gibt eine Reihe von Optionen zum Anpassen einer Warteschlangenkonfiguration. Diese Optionen helfen dir dabei, je nach den Bedürfnissen deines Spiels die bestmöglichen Platzierungen vorzunehmen. Weitere Informationen zum Anpassen von Warteschlangen für dein Spiel findest du unter [Passen Sie eine Warteschlange für Spielsitzungen an](#). Du kannst die meisten Einstellungen für die Warteschlangenkonfiguration jederzeit aktualisieren.

Sie können eine Warteschlange für Spielsitzungen entweder mit der Amazon GameLift Servers Konsole oder der AWS CLI erstellen.

Console

Wählen Sie in der [Amazon GameLift ServersKonsole](#) eine AWS Region aus, in der Sie arbeiten möchten. Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie Warteschlangen.

1. Wählen Sie auf der Seite Warteschlangen die Option Warteschlange erstellen aus, um den Workflow zu starten.
2. Geben Sie unter Warteschlangeneinstellungen die folgenden Einstellungen ein:
 - a. Geben Sie einen Namen für die Warteschlange ein. Dieser Name muss für den Namen, in dem Sie AWS-Region die Warteschlange erstellen, eindeutig sein.
 - b. Behalten Sie die Standardeinstellung für das Timeout bei, die 600 Sekunden (oder 10 Minuten) beträgt. Dieser Wert steuert, wie lange Amazon GameLift Servers versucht wird, eine neue Spielsitzung zu starten, bevor sie beendet wird. Amazon GameLift Servers sucht nach verfügbaren Ressourcen, bis das Zeitlimit für die Anfrage überschritten wird. Sie können die Timeout-Einstellung einer Warteschlange jederzeit aktualisieren.
 - c. Überspringen Sie den Abschnitt Richtlinien zur Player-Latenz. Eine Warteschlange verwendet Latenzrichtlinien nur, wenn sie Platzierungsanfragen erhält, die Latenzdaten von Spielern enthalten. Sie können einer Warteschlange jederzeit Latenzrichtlinien hinzufügen. Weitere Informationen zum Erstellen von Latenzrichtlinien finden Sie unter [Erstellen Sie eine Latenzrichtlinie für Spieler](#).
3. Überspringen Sie den Abschnitt „Orte der Platzierung von Spielsitzungen“, um die Standardeinstellung „Alle Standorte“ zu verwenden. Mit dieser Einstellung kannst du eine Zulassungsliste mit Orten erstellen, an denen die Warteschlange Platzierungen vornehmen

kann (auch als Filterkonfiguration bezeichnet). Weitere Informationen zur Priorisierung nach Standort und Filterkonfigurationen finden Sie unter. [Priorisieren Sie Platzierungen nach Standort](#)

4. Fügen Sie unter Zielreihenfolge eine oder mehrere Flotten zur Warteschlange hinzu. Sie können Flotten mithilfe von fleet IDs oder oder oder ARNs mithilfe eines Flottenalias identifizieren. Wenn du mehrere Flotten hinzufügst, denke daran, dass auf allen Flotten ähnliche Spielversionen ausgeführt werden sollten und dass sie mit jedem Spielclient kompatibel sein sollten, der diese Warteschlange verwendet. Außerdem müssen alle Flotten in einer Warteschlange dieselbe Zertifikatskonfiguration haben.
 - a. Wählen Sie die Region aus, in der die Flotte oder der Alias erstellt wurde. Bei einer Flotte mit mehreren Standorten ist dies die „Heimatregion“.
 - b. Wählen Sie als Zieltyp entweder eine Flotte oder einen Alias aus.
 - c. Ihre Region und Ihre Typauswahl füllen eine Dropdownliste mit vorhandenen Flotten oder Aliasnamen auf. Wählen Sie eines aus, das Sie als Ziel für die Warteschlange festlegen möchten.
 - d. Um eine andere Flotte oder einen anderen Alias für die Warteschlange anzugeben, wählen Sie Ziel hinzufügen und wiederholen Sie die vorherigen Schritte.
 - e. Nachdem Sie eine Liste von Zielen hinzugefügt haben, verwenden Sie die drag-and-drop Funktion, um die Ziele neu anzuordnen. Amazon GameLift Servers verwendet diese Reihenfolge, wenn Platzierungen nach Ziel priorisiert werden.
5. Überspringe den Abschnitt „Priorität der Platzierung von Spielsitzungen“, um die standardmäßige Prioritätsreihenfolge beizubehalten. Mit dieser Einstellung können Sie festlegen, wie Amazon GameLift Servers ausgewählt wird, wo nach verfügbaren Hosting-Ressourcen für neue Platzierungen von Spielsitzungen gesucht werden soll. Weitere Informationen zur Priorisierung von Platzierungen finden Sie unter. [Priorisieren Sie die Platzierung von Spielsitzungen](#) Sie können die Platzierungsprioritäten einer Warteschlange jederzeit aktualisieren.
6. Behalten Sie unter Reihenfolge der Standorte die Standardwerte bei. Diese Einstellung wird bei der Priorisierung nach Flottenstandort verwendet. Sie gibt die Reihenfolge der zu verwendenden Standorte an. Bei Verwendung der standardmäßigen Prioritätseinstellungen wird der Standort als Entscheidungsgrundlage verwendet, wenn das bevorzugte Ziel eine Flotte mit mehreren Standorten ist.
7. Überspringen Sie den optionalen Abschnitt mit den Einstellungen für die Ereignisbenachrichtigung. Für Warteschlangen, die eine große Anzahl von

Platzierungsanfragen bearbeiten, sind Ereignisbenachrichtigungen erforderlich. In Warteschlangen, in denen nur geringe Mengen verarbeitet werden, z. B. zu Entwicklungs- oder Testzwecken, können Sie den Status der Platzierungsanfragen verfolgen, indem Sie eine Abfrage mit durchführen. [DescribeGameSessionPlacement](#) Weitere Details finden Sie unter [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#). Sie können die Einstellungen für die Benachrichtigung über Ereignisse in einer Warteschlange jederzeit aktualisieren.

8. Wählen Sie Erstellen, um eine neue Warteschlange mit minimalen Anpassungen zu erstellen.

AWS CLI

Example Erstellen einer Warteschlange

Im folgenden Beispiel wird eine Warteschlange für Spielsitzungen mit diesen Konfigurationen erstellt:

- Ein Timeout von fünf Minuten.
- Zwei Flottenziele.
- Filtern Sie, um nur Platzierungen an diesen Standorten zuzulassen:us-east-1,us-east-2, us-west-2, und. ca-central-1
- Prioritätsreihenfolge auf der Grundlage der Kosten und dann der Standorte in einer bestimmten Reihenfolge.

```
aws gamelift create-game-session-queue \  
  --name "sample-test-queue" \  
  --timeout-in-seconds 300 \  
  --destinations DestinationArn="arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-772266ba-8c82-4a6e-b620-a74a62a93ff8" DestinationArn="arn:aws:gamelift:us-  
east-1:111122223333:fleet/fleet-33f28fb6-aa8b-4867-85b4-ceb217bf5994" \  
  --filter-configuration "AllowedLocations=us-east-1, ca-central-1, us-east-2, us-  
west-2" \  
  --priority-configuration PriorityOrder="COST","LOCATION",LocationOrder="us-  
east-1","us-east-2","ca-central-1","us-west-2" \  
  --notification-target "arn:aws:sns:us-east-1:111122223333:gamelift-test.fifo"
```

 Note

Sie können ARN-Werte für Flotte und Alias abrufen, indem Sie entweder [describe-fleet-attributes](#) oder [describe-alias](#) mit der Flotte- oder Alias-ID aufrufen.

Wenn die `create-game-session-queue` Anfrage erfolgreich ist, wird ein [GameSessionQueue](#) Objekt mit der neuen Warteschlangenkonfiguration Amazon GameLift Servers zurückgegeben. Sie können jetzt Anfragen an die Warteschlange senden mit [StartGameSessionPlacement](#).

Example Erstellen Sie eine Warteschlange mit Latenzrichtlinien für Spieler

Im folgenden Beispiel wird eine Warteschlange für Spielsitzungen mit diesen Konfigurationen erstellt:

- Ein Timeout von zehn Minuten
- Drei Flottenziele
- Eine Reihe von Richtlinien für die Latenz von Spielern

```
aws gamelift create-game-session-queue \  
  --name "matchmaker-queue" \  
  --timeout-in-seconds 600 \  
  --destinations DestinationArn=arn:aws:gamelift:us-east-1::alias/alias-a1234567-  
b8c9-0d1e-2fa3-b45c6d7e8910 \  
                  DestinationArn=arn:aws:gamelift:us-west-2::alias/alias-b0234567-  
c8d9-0e1f-2ab3-c45d6e7f8901 \  
                  DestinationArn=arn:aws:gamelift:us-west-2::fleet/fleet-f1234567-  
b8c9-0d1e-2fa3-b45c6d7e8912 \  
  --player-latency-policies  
    "MaximumIndividualPlayerLatencyMilliseconds=50,PolicyDurationSeconds=120" \  
  
    "MaximumIndividualPlayerLatencyMilliseconds=100,PolicyDurationSeconds=120" \  
    "MaximumIndividualPlayerLatencyMilliseconds=150" \  

```

Wenn die `create-game-session-queue` Anfrage erfolgreich ist, wird ein [GameSessionQueue](#) Objekt mit der neuen Warteschlangenkonfiguration Amazon GameLift Servers zurückgegeben.

Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten

Du kannst Event-Benachrichtigungen verwenden, um den Status einzelner Platzierungsanfragen zu überwachen. Wir empfehlen, Event-Benachrichtigungen für alle Spiele mit hohem Platzierungsvolumen einzurichten.

Es gibt zwei Optionen für die Einrichtung von Event-Benachrichtigungen.

- Lassen Amazon GameLift Servers Sie mithilfe einer Warteschlange Ereignisbenachrichtigungen zu einem Amazon Simple Notification Service (Amazon SNS) -Thema veröffentlichen.
- Verwenden Sie automatisch veröffentlichte EventBridge Amazon-Ereignisse und die zugehörigen Tools für die Verwaltung von Veranstaltungen.

Eine Liste der Ereignisse zur Platzierung von Spielsitzungen, die von ausgelöst wurden Amazon GameLift Servers, finden Sie unter [Platzierungsveranstaltungen für Spielsitzungen](#).

Important

Für Vermittlungssysteme mit hohem Volumen empfehlen wir, anstelle von FIFO-Themen die Standardthemen (ohne FIFO) von Amazon SNS zu verwenden. Für FIFO-Themen gelten niedrigere Veröffentlichungslimits als für Standardthemen, was bei hoher Auslastung zu eingeschränkten Ausnahmen führen kann. Wenn Sie bei FIFO-Themen eine Drosselung feststellen, verlieren Sie möglicherweise die Benachrichtigungen zur Platzierung in der Warteschlange.

Richten Sie ein SNS-Thema ein

Amazon GameLift Servers Um alle Ereignisse, die durch eine Warteschlange für eine Spielsitzung generiert wurden, zu einem Thema zu veröffentlichen, setzen Sie das Zielfeld für die Benachrichtigung auf ein Thema.

Einrichten eines SNS-Themas für die Amazon GameLift Servers Ereignisbenachrichtigung

1. Melden Sie sich bei <https://console.aws.amazon.com/sns/v3/home> an AWS-Managementkonsole und öffnen Sie die Amazon SNS SNS-Konsole.
2. Wählen Sie auf der Seite mit den SNS-Themen die Option Thema erstellen aus und folgen Sie den Anweisungen, um Ihr Thema zu erstellen.

3. Gehen Sie unter Zugriffsrichtlinie wie folgt vor:
 - a. Wählen Sie die Methode Erweitert.
 - b. Fügen Sie der vorhandenen Richtlinie den folgenden fett gedruckten Abschnitt des JSON-Objekts hinzu.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:111122223333:your_topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "your_account"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "sns:Publish",
```

```

    "Resource": "arn:aws:sns:us-east-1:111122223333:your_topic_name",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:gamelift:us-
east-1:111122223333:gamesessionqueue/your_queue_name"
      }
    }
  }
]
}

```

- c. (Optional) Fügen Sie dem Thema zusätzliche Zugriffskontrolle hinzu, indem Sie der Ressourcenrichtlinie Bedingungen hinzufügen.
4. Wählen Sie Thema erstellen aus.
5. Nachdem Sie Ihr SNS-Thema erstellt haben, fügen Sie es während der Warteschlangenerstellung zu den Warteschlangen hinzu, oder bearbeiten Sie eine bestehende Warteschlange, um es hinzuzufügen.

Richten Sie ein SNS-Thema mit serverseitiger Verschlüsselung ein

Mit der serverseitigen Verschlüsselung (Server-Side Encryption, SSE) können Sie vertrauliche Daten in verschlüsselten Themen speichern. SSE schützt den Inhalt von Nachrichten in Amazon SNS-Themen mit Schlüsseln, die in AWS Key Management Service (AWS KMS) verwaltet werden. Weitere Informationen zur serverseitigen Verschlüsselung mit Amazon SNS finden Sie unter [Verschlüsselung im Ruhezustand im Amazon Simple Notification Service Developer Guide](#).

Um ein SNS-Thema mit serverseitiger Verschlüsselung einzurichten, lesen Sie sich die folgenden Themen durch:

- [Schlüssel erstellen](#) im Entwicklerhandbuch AWS Key Management Service
- [SSE für ein Thema im Amazon Simple Notification Service Developer Guide aktivieren](#)

Verwenden Sie beim Erstellen Ihres KMS-Schlüssels die folgende KMS-Schlüsselrichtlinie:

```

{
  "Effect": "Allow",
  "Principal": {
    "Service": "gamelift.amazonaws.com"
  },

```

```

    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn":
"arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
"arn:aws:sns:your_region:your_account:your_sns_topic_name"
        }
    }
}

```

Einrichten EventBridge

Amazon GameLift Servers veröffentlicht automatisch alle Platzierungsereignisse für Spielsitzungen auf EventBridge. Mit können EventBridge Sie Regeln einrichten, nach denen Ereignisse zur Verarbeitung an Ziele weitergeleitet werden. Sie können beispielsweise eine Regel festlegen, um das Ereignis an eine AWS Lambda Funktion weiterzuleiten `PlacementFulfilled`, die Aufgaben erledigt, die vor der Verbindung zu einer Spielsitzung ausgeführt werden. Weitere Informationen zu EventBridge finden Sie unter [Was ist Amazon EventBridge?](#) im EventBridge Amazon-Benutzerhandbuch.

Im Folgenden finden Sie einige Beispiele für EventBridge Regeln, die mit Amazon GameLift Servers Warteschlangen verwendet werden können:

Entspricht Ereignissen aus allen Warteschlangen Amazon GameLift Servers

```

{
    "source": [
        "aws.gamelift"
    ],
    "detail-type": [
        "GameLift Queue Placement Event"
    ]
}

```

Findet Ereignisse aus einer bestimmten Warteschlange

```
{
  "source": [
    "aws.gamelift"
  ],
  "detail-type": [
    "GameLift Queue Placement Event"
  ],
  "resources": [
    "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
  ]
}
```

Referenzimplementierung mit AWS CDK

Eine vollständige Referenzimplementierung der ereignisbasierten Platzierung von Spielsitzungen mit Amazon SNS und Amazon DynamoDB finden Sie in der [Anleitung zur ereignisbasierten Platzierung von Spielsitzungen](#) im Amazon Toolkit. AWS Lambda GameLift Diese Anleitung umfasst eine AWS CDK Vorlage für die Bereitstellung eines vollständigen ereignisgesteuerten Platzierungssystems für Spielsitzungen mit einer AWS Lambda Funktion zur Verarbeitung von Platzierungsereignissen für Spielsitzungen, eine Amazon DynamoDB-Tabelle zur Verfolgung des Platzierungsstatus von Spielsitzungen und bewährte Methoden für die Warteschlangenkonfiguration. Amazon GameLift Servers

Passen Sie es an Ihre Game-Hosting-Lösung an

Wenn eine grundlegende Game-Hosting-Lösung eingerichtet ist, können Sie diese anhand der folgenden Themen anpassen und erweitern, um das Spielerlebnis zu verbessern, die Kosten zu optimieren und erweiterte Funktionen hinzuzufügen. In diesem Abschnitt werden verschiedene Anpassungsoptionen behandelt, die nach der Komponente geordnet sind, auf die sie sich hauptsächlich auswirken. Wählen Sie die Anpassungen, die den Anforderungen und der Spielerbasis Ihres Spiels am besten entsprechen.

Topics

- [Anpassungen des Gameserver-Builds](#)
 - [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#)
 - [Lass deinen Gameserver auf Amazon GameLift Servers Flottendaten zugreifen](#)
 - [VPC-Peering einrichten für Amazon GameLift Servers](#)

- [Spietersitzungen und Matchmaking-Anpassungen](#)
 - [Spieler generieren IDs](#)
 - [Fügen Sie FlexMatch Matchmaking hinzu zu Amazon GameLift Servers](#)
- [Anpassungen der Platzierung von Spielsitzungen](#)
 - [Passen Sie eine Warteschlange für Spielsitzungen an](#)
 - [Priorisieren Sie die Platzierung von Spielsitzungen](#)
 - [Eine Warteschlange für Spot-Instances erstellen](#)
- [Anpassungen der Hosting-Ressourcen](#)
 - [Wählen Sie Rechenressourcen für eine verwaltete Flotte](#)
 - [Passen Sie eine Amazon GameLift Servers Containerflotte an](#)
 - [Senken Sie die Kosten für das Hosting von Spielen mit Spot-Flotten](#)
 - [Optimiere die Laufzeitkonfiguration des Spieleservers auf verwalteten Amazon GameLift Servers](#)
 - [Arbeite mit dem Amazon GameLift Servers Agenten](#)
 - [Zusammenfassung einer Amazon GameLift Servers Flottenbezeichnung mit einem Alias](#)

Anpassungen des Gameserver-Builds

Anpassungen beim Aufbau von Spieleservern bieten Möglichkeiten, die Funktionalität des Spieleservers zu erweitern, einschließlich der Nutzung anderer AWS Dienste. Diese Anpassungen erweitern die Funktionen Ihres Spieleservers über das grundlegende Hosting hinaus und unterstützen erweiterte Funktionen und Integrationen.

Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen

Wenn Sie einen Spieleserver-Build für den Einsatz auf Amazon GameLift Servers Flotten erstellen, möchten Sie vielleicht, dass die Anwendungen in Ihrem Spiel-Build direkt und sicher mit anderen AWS Ressourcen kommunizieren, die Sie besitzen. Da Ihre Spiele-Hosting-Flotten Amazon GameLift Servers verwaltet werden, müssen Sie Amazon GameLift Servers eingeschränkten Zugriff auf diese Ressourcen und Dienste gewähren.

Einige Beispielszenarien umfassen:

- Verwenden Sie einen CloudWatch Amazon-Agenten, um Metriken, Protokolle und Traces von verwalteten EC2-Flotten und Anywhere-Flotten zu sammeln.

- Senden Sie Instanzprotokolldaten an Amazon CloudWatch Logs.
- Besorgen Sie sich Spieldateien, die in einem Amazon Simple Storage Service (Amazon S3) - Bucket gespeichert sind.
- Lesen und Schreiben von Spieldaten (wie Spielmodi oder Inventar), die in einer Amazon DynamoDB Datenbank oder einem anderen Datenspeicherdienst gespeichert sind.
- Senden Sie mithilfe von Amazon Simple Queue Service (Amazon SQS) Signale direkt an eine Instance.
- Greifen Sie auf benutzerdefinierte Ressourcen zu, die auf Amazon Elastic Compute Cloud (Amazon EC2) bereitgestellt und ausgeführt werden.

Amazon GameLift Servers unterstützt die folgenden Methoden zur Einrichtung des Zugriffs:

- [Greifen Sie mit einer IAM-Rolle auf AWS Ressourcen zu](#)
- [Greifen Sie mit VPC-Peering auf AWS Ressourcen zu](#)

Greifen Sie mit einer IAM-Rolle auf AWS Ressourcen zu

Verwenden Sie eine IAM-Rolle, um anzugeben, wer auf Ihre Ressourcen zugreifen kann, und um Beschränkungen für diesen Zugriff festzulegen. Vertrauenswürdige Parteien können eine Rolle „übernehmen“ und temporäre Sicherheitsanmeldedaten erhalten, die sie zur Interaktion mit den Ressourcen berechtigen. Wenn die Parteien API-Anfragen im Zusammenhang mit der Ressource stellen, müssen sie die Anmeldeinformationen angeben.

Gehen Sie wie folgt vor, um den Zugriff einzurichten, der durch eine IAM-Rolle gesteuert wird:

1. [Erstellen Sie die IAM-Rolle](#)
2. [Ändern Sie Anwendungen, um Anmeldeinformationen abzurufen](#)
3. [Ordnen Sie der IAM-Rolle eine Flotte zu](#)

Erstellen Sie die IAM-Rolle

In diesem Schritt erstellen Sie eine IAM-Rolle mit einer Reihe von Berechtigungen zur Steuerung des Zugriffs auf Ihre AWS Ressourcen und einer Vertrauensrichtlinie, die Amazon GameLift Servers Rechte zur Nutzung der Rollenberechtigungen gewährt.

Anweisungen zum Einrichten der IAM-Rolle finden Sie unter. [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#) Wählen Sie bei der Erstellung der Berechtigungsrichtlinie bestimmte

Dienste, Ressourcen und Aktionen aus, mit denen Ihre Anwendungen funktionieren müssen. Es hat sich bewährt, den Umfang der Berechtigungen so weit wie möglich einzuschränken.

Nachdem Sie die Rolle erstellt haben, notieren Sie sich den Amazon-Ressourcennamen (ARN) der Rolle. Sie benötigen die Rolle ARN bei der Flottenerstellung.

Ändern Sie Anwendungen, um Anmeldeinformationen abzurufen

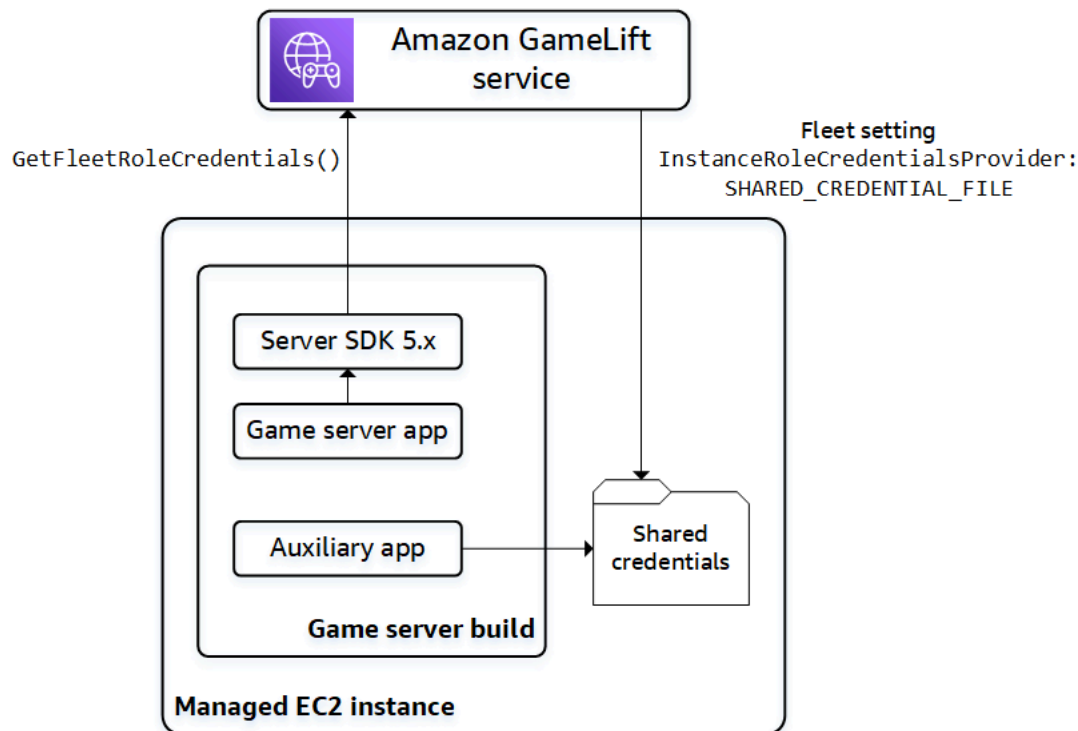
In diesem Schritt konfigurieren Sie Ihre Anwendungen so, dass sie Sicherheitsanmeldeinformationen für die IAM-Rolle abrufen und diese bei der Interaktion mit Ihren AWS Ressourcen verwenden. In der folgenden Tabelle können Sie anhand (1) des Anwendungstyps und (2) der Server-SDK-Version, mit Amazon GameLift Servers der Ihr Spiel kommuniziert, bestimmen, wie Sie Ihre Anwendungen ändern können.

	Spieleserver-Anwendungen	Andere Anwendungen
Verwenden Sie das Server-SDK Version 5.x	Rufen Sie die Server-SDK-Methode <code>GetFleetRoleCredentials()</code> von Ihrem Spieleservercode aus auf.	Fügen Sie der Anwendung Code hinzu, um Anmeldeinformationen aus einer gemeinsam genutzten Datei auf der Flotteninstanz abzurufen.
Verwenden Sie das Server-SDK Version 4 oder früher	Call AWS -Security-Token-Service (AWS STS) AssumeRole mit der Rolle ARN.	Call AWS -Security-Token-Service (AWS STS) AssumeRole mit der Rolle ARN.

Note

Bei Containerflotten werden `FleetRoleArn` Anmeldeinformationen in jeden Container eingefügt. Ihre Anwendungen können den AWS Standardanmeldedienstanbieter verwenden, um auf diese Anmeldeinformationen zuzugreifen. Sie können immer noch `anrufenGetFleetRoleCredentials()`, wodurch dieselben Anmeldeinformationen zurückgegeben werden. Auf diese Anmeldeinformationen für die Flottenrolle kann nur innerhalb von Containern zugegriffen werden.

Für Spiele, die in das Server-SDK 5.x integriert sind, zeigt dieses Diagramm, wie Anwendungen in Ihrem bereitgestellten Spiel-Build Anmeldeinformationen für die IAM-Rolle abrufen können.



Rufen Sie an **GetFleetRoleCredentials()** (Server-SDK 5.x)

Rufen Sie in Ihrem Spielservercode, der bereits in das Amazon GameLift Servers Server-SDK 5.x integriert sein sollte, `GetFleetRoleCredentials` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) auf, um einen Satz temporärer Anmeldeinformationen abzurufen. Wenn die Anmeldeinformationen ablaufen, kannst du sie mit einem weiteren Aufruf von `aktualisieren`. `GetFleetRoleCredentials`

Verwenden Sie gemeinsam genutzte Anmeldeinformationen (Server-SDK 5.x)

Fügen Sie für Nicht-Serveranwendungen, die mit Spielserver-Builds mithilfe des Server-SDK 5.x bereitgestellt werden, Code hinzu, um die in einer gemeinsam genutzten Datei gespeicherten Anmeldeinformationen abzurufen und zu verwenden. Amazon GameLift Servers generiert ein Anmeldeinformationsprofil für jede Flotteninstanz. Die Anmeldeinformationen können von allen Anwendungen auf der Instance verwendet werden. Amazon GameLift Servers aktualisiert kontinuierlich die temporären Anmeldeinformationen.

Sie müssen eine Flotte so konfigurieren, dass die Datei mit den gemeinsamen Anmeldeinformationen bei der Flottenerstellung generiert wird.

Geben Sie in jeder Anwendung, die die Datei mit den gemeinsamen Anmeldeinformationen verwenden muss, den Speicherort und den Profilnamen wie folgt an:

Windows:


```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "C:\\\\Credentials\\\\credentials"
```

Linux:

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "/local/credentials/credentials"
```

Beispiel: Richten Sie einen CloudWatch Agenten ein, der Metriken für Amazon GameLift Servers Flotteninstanzen sammelt

Wenn Sie einen CloudWatch Amazon-Agenten verwenden möchten, um Kennzahlen, Protokolle und Rückverfolgungen von Ihren Amazon GameLift Servers Flotten zu sammeln, verwenden Sie diese Methode, um den Agenten zu autorisieren, die Daten an Ihr Konto zu senden. Führen Sie in diesem Szenario die folgenden Schritte aus:

1. Rufen Sie die CloudWatch `config.json` Agentendatei ab oder schreiben Sie sie.
2. Aktualisieren Sie die `common-config.toml` Datei für den Agenten, um den Namen der Anmeldeinformationsdatei und den Profilnamen wie oben beschrieben zu identifizieren.
3. Richten Sie das Build-Installationsskript Ihres Gameservers ein, um den CloudWatch Agenten zu installieren und zu starten.

Verwenden Sie **AssumeRole()** (Server-SDK 4)

Fügen Sie Ihren Anwendungen Code hinzu, um die IAM-Rolle zu übernehmen und Anmeldeinformationen für die Interaktion mit Ihren AWS Ressourcen zu erhalten. Jede Anwendung, die auf einer Amazon GameLift Servers Flotteninstanz mit Server-SDK 4 oder früher ausgeführt wird, kann die IAM-Rolle übernehmen.

Im Anwendungscode muss die Anwendung vor dem Zugriff auf eine AWS Ressource die [AssumeRole](#) API-Operation AWS -Security-Token-Service (AWS STS) aufrufen und den Rollen-ARN angeben. Dieser Vorgang gibt einen Satz temporärer Anmeldeinformationen zurück, die die Anwendung zum Zugriff auf die AWS Ressource autorisieren. Weitere Informationen finden Sie unter [Verwenden temporärer Anmeldeinformationen mit AWS Ressourcen](#) im IAM-Benutzerhandbuch.

Ordnen Sie der IAM-Rolle eine Flotte zu

Nachdem du die IAM-Rolle erstellt und die Anwendungen in deinem Gameserver-Build aktualisiert hast, um die Zugangsdaten abzurufen und zu verwenden, kannst du eine Flotte bereitstellen. Wenn du die neue Flotte konfigurierst, lege die folgenden Parameter fest.

Für Containerflotten:

- [FleetRoleArn](#)— Setzen Sie diesen Parameter auf den ARN der IAM-Rolle.

Für andere Flottenarten:

- [InstanceRoleArn](#)— Setzen Sie diesen Parameter auf den ARN der IAM-Rolle.
- [InstanceRoleCredentialsProvider](#)— Um zur Generierung einer gemeinsamen Anmeldeinformationsdatei für jede Flotteninstanz aufzufordernAmazon GameLift Servers, setzen Sie diesen Parameter aufSHARED_CREDENTIAL_FILE.

Sie müssen diese Werte festlegen, wenn Sie die Flotte erstellen. Sie können später nicht aktualisiert werden.

Greifen Sie mit VPC-Peering auf AWS Ressourcen zu

Sie können Amazon Virtual Private Cloud (Amazon VPC) -Peering verwenden, um zwischen Anwendungen, die auf einer Amazon GameLift Servers Instance ausgeführt werden, und einer anderen AWS Ressource zu kommunizieren. Eine VPC ist ein von Ihnen definiertes virtuelles privates Netzwerk, das eine Reihe von Ressourcen umfasst, die über Ihr AWS-Konto verwaltet werden. Jede Amazon GameLift Servers-Flotte hat eine eigene VPC. Mit VPC-Peering können Sie eine direkte Netzwerkverbindung zwischen der VPC für Ihre Flotte und für Ihre anderen Ressourcen herstellen.

AWS

Amazon GameLift Servers optimiert den Prozess der Einrichtung von VPC-Peering-Verbindungen für Ihre Spieleserver. Es übernimmt Peering-Anfragen, aktualisiert Routing-Tabellen und konfiguriert die Verbindungen nach Bedarf. Anweisungen zum Einrichten von VPC-Peering für Ihre Spieleserver finden Sie unter. [VPC-Peering einrichten für Amazon GameLift Servers](#)

 Note

VPC-Peering wird für Containerflotten nicht unterstützt.

Lass deinen Gameserver auf Amazon GameLift Servers Flottendaten zugreifen

Es gibt Situationen, in denen dein benutzerdefinierter Spiel-Build oder dein benutzerdefiniertes Amazon GameLift Servers Realtime Skript möglicherweise Informationen über die Flotte benötigt. Amazon GameLift Servers Ihr Spiel-Build oder Ihr Skript könnte beispielsweise Code für Folgendes enthalten:

- Überwachen Sie die Aktivitäten auf der Grundlage von Flottendaten.
- Führen Sie Kennzahlen zusammen, um die Aktivitäten anhand von Flottendaten zu verfolgen. (Viele Spiele verwenden diese Daten für LiveOps Aktivitäten.)
- Stellen Sie relevante Daten für benutzerdefinierte Spieledienste bereit, z. B. für Matchmaking, zusätzliche Kapazitätsskalierung oder Tests.

Flotteninformationen sind als JSON-Datei für jede Instanz an den folgenden Speicherorten verfügbar:

- Windows: C:\GameMetadata\gamelift-metadata.json
- Linux: /local/gamemetadata/gamelift-metadata.json

Die `gamelift-metadata.json` Datei enthält die [Attribute einer Amazon GameLift Servers Flottenressource](#).

Beispiel für eine JSON-Datei:

```
{
  "buildArn": "arn:aws:gamelift:us-west-2:123456789012:build/build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "buildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "fleetArn": "arn:aws:gamelift:us-west-2:123456789012:fleet/fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "fleetDescription": "Test fleet for Really Fun Game v0.8",
  "fleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "name": "ReallyFunGameTestFleet08",
  "fleetType": "ON_DEMAND",
  "instanceRoleArn": "arn:aws:iam::123456789012:role/S3AccessForGameLift",
  "instanceType": "c5.large",
  "serverLaunchPath": "/local/game/reallyfungame.exe"
}
```

VPC-Peering einrichten für Amazon GameLift Servers

Dieses Thema enthält Anleitungen zum Einrichten einer VPC-Peering-Verbindung zwischen Ihren über Amazon GameLift Servers gehosteten Spielservern und Ihren anderen Nicht-Amazon GameLift Servers Ressourcen. Verwenden Sie Amazon Virtual Private Cloud (VPC) -Peering-Verbindungen, damit Ihre Spielserver direkt und privat mit Ihren anderen AWS Ressourcen wie einem Webservice oder einem Repository kommunizieren können. Sie können VPC-Peering mit allen Ressourcen einrichten, die auf einem AWS Konto laufen AWS und von diesem verwaltet werden, auf das Sie Zugriff haben.

Note

VPC-Peering ist eine erweiterte Funktion. Informationen zu den bevorzugten Optionen, mit denen Sie Ihren Spielservern die direkte und private Kommunikation mit Ihren anderen AWS Ressourcen ermöglichen, finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spielserver mit anderen AWS Ressourcen](#)

Wenn Sie bereits mit Amazon VPCs und VPC-Peering vertraut sind, sollten Sie sich darüber im Klaren sein, dass die Einrichtung von Peering mit Amazon GameLift Servers Spielservern etwas anders ist. Sie haben keinen Zugriff auf die VPC, die Ihre Spielserver enthält — sie wird vom Amazon GameLift Servers Dienst gesteuert —, sodass Sie nicht direkt VPC-Peering für sie anfordern können. Stattdessen autorisieren Sie den VPC zunächst mit Ihren Nicht-Amazon GameLift Servers-Ressourcen, eine Peering-Anfrage vom Amazon GameLift Servers-Service anzunehmen. Dann starten Sie Amazon GameLift Servers, um das VPC-Peering anzufordern, das Sie gerade autorisiert haben. Amazon GameLift Servers übernimmt die Aufgaben zum Erstellen der Peering-Verbindung, zum Einrichten der Routentabellen und zum Konfigurieren der Verbindung.

So richten Sie VPC-Peering für eine bestehende Flotte ein

1. Besorgen Sie sich Konto-ID AWS (s) und Anmeldeinformationen.

Sie benötigen eine ID und Anmeldeinformationen für die folgenden AWS Konten. Sie können das AWS Konto finden, IDs indem Sie sich bei anmelden [AWS-Managementkonsole](#) und Ihre Kontoeinstellungen einsehen. Die Anmeldeinformationen können Sie von der IAM-Konsole abrufen.

- AWS Konto, mit dem du deine Amazon GameLift Servers Spielserver verwaltest.

- AWS Konto, das du zur Verwaltung deiner Amazon GameLift Servers Nicht-Ressourcen verwendest.

Wenn Sie dasselbe Konto für Amazon GameLift Servers und für Amazon GameLift Servers Nicht-Ressourcen verwenden, benötigen Sie nur eine ID und Anmeldeinformationen für dieses Konto.

2. Kennungen für jede VPC abrufen.

Besorgen Sie sich die folgenden Informationen für die beiden VPCs , die miteinander verbunden werden sollen:

- VPC für deine Amazon GameLift Servers Spielservers — Das ist deine Amazon GameLift Servers Flotten-ID. Ihre Spielservers werden in Amazon GameLift Servers auf einer Flotte von EC2-Instances bereitgestellt. Jede Flotte wird automatisch in ihrer eigenen VPC platziert, die vom Amazon GameLift Servers-Service verwaltet wird. Sie haben keinen direkten Zugriff auf die VPC, daher wird sie anhand der Flotten-ID identifiziert.
- VPC für Ihre Amazon GameLift Servers AWS Nicht-Ressourcen — Sie können ein VPC-Peering mit allen Ressourcen einrichten, die auf einem AWS Konto laufen AWS und von diesem verwaltet werden, auf das Sie Zugriff haben. Wenn Sie noch keine VPC für diese Ressourcen erstellt haben, finden Sie weitere Informationen unter [Erste Schritte mit Amazon VPC](#). Nachdem Sie eine VPC erstellt haben, können Sie die VPC-ID finden, indem Sie sich bei der [AWS-Managementkonsole](#) für Amazon VPC anmelden und Ihre anzeigen. VPCs

Note

Wenn Sie ein Peering einrichten, VPCs müssen beide in derselben Region existieren. Die VPC für Ihre Amazon GameLift Servers-Flottenspielservers befindet sich in derselben Region wie die Flotte.

3. Autorisieren Sie ein VPC-Peering.

In diesem Schritt autorisieren Sie eine zukünftige Anfrage von Amazon GameLift Servers, um zwischen der VPC mit Ihren Spielservers und Ihrer VPC für Nicht-Amazon GameLift Servers-Ressourcen eine Peer-Verbindung herzustellen. Diese Aktion aktualisiert die Sicherheitsgruppe für Ihre VPC.

Um das VPC-Peering zu autorisieren, rufen Sie die Service-API [CreateVpcPeeringAuthorization\(\)](#) auf oder verwenden Sie den AWS CLI-Befehl. `create-vpc-peering-authorization` Führen Sie diesen Anruf über das Konto durch, das Ihre Nicht-Amazon GameLift Servers Ressourcen verwaltet. Geben Sie die folgenden Informationen an:

- Peer-VPC-ID — Dies ist für die VPC mit Ihren Amazon GameLift Servers Nicht-Ressourcen.
- Amazon GameLift Servers AWS Konto-ID — Dies ist das Konto, das Sie zur Verwaltung Ihrer Amazon GameLift Servers Flotte verwenden.

Sobald Sie ein VPC-Peering autorisiert haben, bleibt die Autorisierung 24 Stunden lang gültig, sofern sie nicht widerrufen wird. Sie können Ihre VPC-Peering-Autorisierungen unter Verwendung der folgenden Operationen verwalten:

- [DescribeVpcPeeringAuthorizations\(\)](#) (AWS CLI `describe-vpc-peering-authorizations`).
- [DeleteVpcPeeringAuthorization\(\)](#) (AWS CLI `delete-vpc-peering-authorization`).

4. Fordern Sie eine Peering-Verbindung an.

Mit einer gültigen Autorisierung können Sie das Herstellen einer Peering-Verbindung durch Amazon GameLift Servers anfordern.

Um ein VPC-Peering anzufordern, rufen Sie die Service-API [CreateVpcPeeringConnection\(\)](#) auf oder verwenden Sie den AWS CLI-Befehl. `create-vpc-peering-connection` Führen Sie diesen Anruf mit dem Konto durch, das Ihre Amazon GameLift Servers-Spieler server verwaltet. Verwenden Sie die folgenden Informationen, um die beiden zu identifizieren, für VPCs die Sie eine Peering-Verbindung herstellen möchten:

- Peer-VPC-ID und AWS Konto-ID — Dies ist die VPC für Ihre Amazon GameLift Servers Nicht-Ressourcen und das Konto, mit dem Sie sie verwalten. Die VPC-ID muss mit der ID einer gültigen Peering-Autorisierung übereinstimmen.
- Flotten-ID — Dies identifiziert die VPC für Ihre Amazon GameLift Servers Spieler server.

5. Verfolgen Sie den Status der Peering-Verbindung.

Die Anforderung einer VPC-Peering-Verbindung ist ein asynchroner Vorgang. Um den Status einer Peering-Anfrage nachzuverfolgen und den Erfolg oder das Fehlschlagen zu verarbeiten, verwenden Sie eine der folgenden Optionen:

- Stetige Abfrage mit `DescribeVpcPeeringConnections()`. Diese Operation ruft den Datensatz für die VPC-Peering-Verbindung ab, einschließlich des Status der Anforderung. Wenn eine Peering-Verbindung erfolgreich erstellt wurde, enthält der Verbindungsdatensatz auch einen CIDR-Block mit privaten IP-Adressen, der der VPC zugewiesen wird.
- Behandeln Sie Flottenereignisse im Zusammenhang mit VPC-Peering-Verbindungen mit [DescribeFleetEvents\(\)](#), einschließlich Erfolgs- und Fehlschlagsereignissen.

Sobald die Peering-Verbindung hergestellt wurde, können Sie sie mit den folgenden Vorgängen verwalten:

- [DescribeVpcPeeringConnections\(\)](#) (AWS CLI `describe-vpc-peering-connections`).
- [DeleteVpcPeeringConnection\(\)](#) (AWS CLI `delete-vpc-peering-connection`).

So richten Sie VPC-Peering mit einer neuen Flotte ein

Sie können eine neue Amazon GameLift Servers-Flotte erstellen und gleichzeitig eine VPC-Peering-Verbindung anfordern.

1. Holen Sie sich AWS Konto-ID (s) und Anmeldeinformationen.

Sie benötigen eine ID und Anmeldeinformationen für die folgenden beiden AWS Konten. Sie können das AWS Konto finden, IDs indem Sie sich bei anmelden [AWS-Managementkonsole](#) und Ihre Kontoeinstellungen einsehen. Die Anmeldeinformationen können Sie von der IAM-Konsole aus abrufen.

- AWS Konto, mit dem du deine Amazon GameLift Servers Spieleserver verwaltest.
- AWS Konto, das du zur Verwaltung deiner Amazon GameLift Servers Nicht-Ressourcen verwendest.

Wenn Sie dasselbe Konto für Amazon GameLift Servers und für Amazon GameLift Servers Nicht-Ressourcen verwenden, benötigen Sie nur eine ID und Anmeldeinformationen für dieses Konto.

2. Holen Sie sich die VPC-ID für Ihre Amazon GameLift Servers AWS Nicht-Ressourcen.

Wenn Sie noch keine VPC für diese Ressourcen erstellt haben, tun Sie dies jetzt (siehe [Erste Schritte mit Amazon VPC](#)). Stellen Sie sicher, dass Sie die neue VPC in derselben Region

erstellen, in der Sie Ihre neue Flotte erstellen möchten. Wenn Ihre Amazon GameLift Servers Nicht-Ressourcen unter einem anderen AWS Konto oder einer anderen user/user Gruppe verwaltet werden als dem Amazon GameLift Servers, mit dem Sie sie verwenden, müssen Sie diese Kontoanmeldeinformationen verwenden, wenn Sie im nächsten Schritt die Autorisierung beantragen.

Nachdem Sie eine VPC erstellt haben, können Sie die VPC-ID in der Amazon VPC-Konsole finden, indem Sie Ihre aufrufen. VPCs

3. Autorisieren eines VPC-Peerings mit anderen als Amazon GameLift Servers-Ressourcen.

Wenn Amazon GameLift Servers die neue Flotte und eine entsprechende VPC erstellt, wird auch eine Anfrage für ein Peering an die VPC für Ihre Nicht-Amazon GameLift Servers-Ressourcen gesendet. Sie müssen diese Anfrage vorab autorisieren. Dieser Schritt aktualisiert die Sicherheitsgruppe für Ihre VPC.

Rufen Sie mit den Kontoanmeldedaten, die Ihre Amazon GameLift Servers Nicht-Ressourcen verwalten, die Service-API [CreateVpcPeeringAuthorization\(\)](#) auf oder verwenden Sie den AWS CLI-Befehl `create-vpc-peering-authorization`. Geben Sie die folgenden Informationen an:

- Peer-VPC-ID — ID der VPC mit Ihren Amazon GameLift Servers Nicht-Ressourcen.
- Amazon GameLift Servers AWS Konto-ID — ID des Kontos, das Sie zur Verwaltung Ihrer Amazon GameLift Servers Flotte verwenden.

Sobald Sie ein VPC-Peering autorisiert haben, bleibt die Autorisierung 24 Stunden lang gültig, sofern sie nicht widerrufen wird. Sie können Ihre VPC-Peering-Autorisierungen unter Verwendung der folgenden Operationen verwalten:

- [DescribeVpcPeeringAuthorizations\(\)](#) (AWS CLI `describe-vpc-peering-authorizations`).
- [DeleteVpcPeeringAuthorization\(\)](#) (AWS CLI `delete-vpc-peering-authorization`).

4. Folgen Sie den Anweisungen zum [Erstellen einer neuen Flotte mit der AWS CLI](#). Fügen Sie die folgenden zusätzlichen Parameter ein:

- peer-vpc-aws-account-id — ID für das Konto, das Sie verwenden, um die VPC mit Ihren Amazon GameLift Servers Nicht-Ressourcen zu verwalten.
- peer-vpc-id — ID der VPC mit Ihrem Amazon GameLift Servers Nicht-Konto.

Ein erfolgreicher Aufruf von [create-fleet](#) mit den VPC-Peering-Parametern generiert sowohl eine neue Flotte als auch eine neue VPC-Peering-Anfrage. Der Status der Flotte wird auf New gesetzt, und der Flottenaktivierungsprozess wird veranlasst. Der Status der Peering-Verbindungsanforderung wird auf initiating-request gesetzt. Sie können den Erfolg oder Misserfolg der Peering-Anfrage nachverfolgen, indem Sie anrufen. [describe-vpc-peering-connections](#)

Beim Anfordern einer neuen Flotte und einer VPC-Peering-Verbindung sind beide Aktionen erfolgreich oder schlagen fehl. Wenn eine Flotte während des Erstellungsprozesses fehlschlägt, wird die VPC-Peering-Verbindung nicht hergestellt. Analog dazu gilt, wenn eine VPC-Peering-Verbindung aus irgendeinem Grund fehlschlägt, kann auch die neue Flotte nicht vom Status Activating in den Status Active übergehen.

 Note

Die neue VPC-Peering-Verbindung wird erst hergestellt, wenn die Flotte zur Aktivierung bereit ist. Dies bedeutet, dass die Verbindung nicht verfügbar ist und während der Installation des Spielerservers nicht verwendet werden kann.

Im folgenden Beispiel wird sowohl eine neue Flotte als auch eine Peering-Verbindung zwischen einer vordefinierten VPC und der VPC für die neue Flotte erstellt. Die vorab eingerichtete VPC wird durch die Kombination aus Ihrer Amazon GameLift Servers AWS Nicht-Konto-ID und der VPC-ID eindeutig identifiziert.

```
$ AWS gamelift create-fleet
  --name "My_Fleet_1"
  --description "The sample test fleet"
  --ec2-instance-type "c5.large"
  --fleet-type "ON_DEMAND"
  --build-id "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff"
  --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
                           MaxConcurrentGameSessionActivations=2,
                           ServerProcesses=[{LaunchPath=C:\game\Bin64.dedicated
\MultiplayerSampleProjectLauncher_Server.exe,
                                           Parameters=+sv_port 33435 +start_lobby,
                                           ConcurrentExecutions=10}]"
  --new-game-session-protection-policy "FullProtection"
  --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
                                   PolicyPeriodInMinutes=15"
```

```
--ec2-inbound-permissions
"FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"

"FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP"
--metric-groups "EMEAfleets"
--peer-vpc-aws-account-id "111122223333"
--peer-vpc-id "vpc-a11a11a"
```

Kopierbare Version:

```
AWS gamelift create-fleet --name "My_Fleet_1" --description "The
sample test fleet" --fleet-type "ON_DEMAND" --metric-groups
"EMEAfleets" --build-id "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff"
--ec2-instance-type "c5.large" --runtime-configuration
"GameSessionActivationTimeoutSeconds=300,MaxConcurrentGameSessionActivations=2,ServerProcesses
\game\Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe,Parameters=
+sv_port 33435 +start_lobby,ConcurrentExecutions=10}]" --new-game-session-
protection-policy "FullProtection" --resource-creation-limit-policy
"NewGameSessionsPerCreator=3,PolicyPeriodInMinutes=15" --ec2-inbound-
permissions "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
"FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP" --peer-vpc-aws-account-id
"111122223333" --peer-vpc-id "vpc-a11a11a"
```

Beheben von VPC-Peering-Problemen

Wenn Sie Probleme beim Herstellen einer VPC-Peering-Verbindung für Ihre Amazon GameLift Servers-Spielservers haben, beachten Sie die folgenden häufigsten Ursachen:

- Die Autorisierung für die angeforderte Verbindung wurde nicht gefunden.
 - Überprüfen Sie den Status einer VPC-Autorisierung für die Nicht-Amazon GameLift Servers-VPC. Sie ist möglicherweise nicht vorhanden oder abgelaufen.
 - Überprüfen Sie die Regionen der beiden, die VPCs Sie miteinander verbinden möchten. Wenn sie sich nicht in der gleichen Region befinden, kann keine Peer-Verbindung hergestellt werden.
- Die CIDR-Blöcke (siehe [Ungültige VPC-Peering-Verbindungskonfigurationen](#)) Ihrer beiden VPCs überschneiden sich. Die IPv4 CIDR-Blöcke, die Peered zugewiesen sind, dürfen sich nicht überschneiden. VPCs Der CIDR-Block der VPC für Ihre Amazon GameLift Servers Flotte wird automatisch zugewiesen und kann nicht geändert werden. Sie müssen also den CIDR-Block für die VPC für Ihre Nicht-Ressourcen ändern. Amazon GameLift Servers So beheben Sie dieses Problem
 - Suchen Sie diesen CIDR-Block für Ihre Amazon GameLift Servers-Flotte, indem Sie `DescribeVpcPeeringConnections()` aufrufen.

- Rufen Sie die Amazon-VPC-Konsole auf, suchen Sie die VPC für Ihre Nicht-Amazon GameLift Servers-Ressourcen und ändern Sie den CIDR-Block so, dass sie sich nicht überlappen.
- Die neue Flotte wurde nicht aktiviert (wenn VPC-Peering mit einer neuen Flotte angefordert wurde). Wenn die neue Flotte nicht auf den Status „Aktiv“ gewechselt hat, ist keine VPC vorhanden, mit der eine Peering-Verbindung hergestellt werden kann.

Spiellersitzungen und Matchmaking-Anpassungen

Anpassungen an Spiellersitzungen und Spielerzuweisungen geben dir die Möglichkeit, ausgeklügelte Workflows für die Spielerverwaltung zu entwickeln. Dazu gehört auch ein differenziertes Matchmaking-System, das dir dabei hilft, ausgewogene und fesselnde Mehrspielererlebnisse zu bieten.

Spieler generieren IDs

Amazon GameLift Servers verwendet eine Spiellersitzung, um einen Spieler darzustellen, der mit einer Spielsitzung verbunden ist. Amazon GameLift Servers erstellt jedes Mal, wenn ein Spieler über einen integrierten Spielclient eine Verbindung zu einer Spielsitzung herstellt, eine Spiellersitzung Amazon GameLift Servers. Wenn ein Spieler ein Spiel verlässt, endet die Spiellersitzung. Amazon GameLift Servers verwendet Spiellersitzungen nicht wieder.

Important

Wenn du die FlexMatch Spielerzuweisung verwendest und eine neue Matchmaking-Anfrage erstellst, die eine Spieler-ID enthält, die bereits in einer bestehenden aktiven Spielerzuweisungsanfrage enthalten ist, wird die bestehende Anfrage automatisch storniert. Für die stornierte MatchmakingCancelled Anfrage wird jedoch kein Event gesendet. Um den Status vorhandener Matchmaking-Anfragen zu überwachen, können Sie [DescribeMatchmaking](#) den Status der Anfrage in seltenen Intervallen (30-60 Sekunden) abfragen. Für die stornierte Anfrage wird der Status CANCELLED mit dem Grund angezeigt. Cancelled due to duplicate player

Das folgende Codebeispiel generiert zufällig einen eindeutigen Spieler IDs:

```
bool includeBrackets = false;  
bool includeDashes = true;
```

```
string playerId = AZ::Uuid::CreateRandom().ToString<string>(includeBrackets,  
includeDashes);
```

Weitere Informationen zu Spielsitzungen finden Sie unter [Spiel- und Spielsitzungen in der Amazon GameLift Servers Konsole](#).

Fügen Sie FlexMatch Matchmaking hinzu zu Amazon GameLift Servers

Verwenden Sie Amazon GameLift Servers FlexMatch diese Option, um Ihren Amazon GameLift Servers gehosteten Spielen Funktionen zur Spielsuche hinzuzufügen. Sie können es entweder FlexMatch mit benutzerdefinierten Spielserversn oder Amazon GameLift Servers Realtime verwenden.

FlexMatch verbindet den Matchmaking-Service mit einer anpassbaren Regel-Engine. Sie entwerfen anhand von Spielerattributen und Spielmodi, die für Ihr Spiel sinnvoll sind, wie Spieler zusammengebracht werden. FlexMatch verwaltet die Einzelheiten der Bewertung von Spielern, die nach einem Spiel suchen, der Organisation von Spielen mit einem oder mehreren Teams und dem Starten von Spielsitzungen, um die Spiele auszurichten.

Um den vollen FlexMatch Service nutzen zu können, müssen Sie Ihre Hosting-Ressourcen mit Warteschlangen eingerichtet haben. Amazon GameLift Servers verwendet Warteschlangen, um die bestmöglichen Hosting-Standorte für Spiele in verschiedenen Regionen und Computertypen zu finden. Insbesondere können Amazon GameLift Servers Warteschlangen Latenzdaten verwenden, sofern sie von Spielclients bereitgestellt werden, um Spielsitzungen so zu platzieren, dass Spieler beim Spielen die geringstmögliche Latenz erleben.

Weitere Informationen dazu, FlexMatch wie Sie detaillierte Hilfe bei der Integration von Spielsuche in Ihre Spiele erhalten, finden Sie in den folgenden Themen im [Amazon GameLift Servers FlexMatch Entwicklerhandbuch](#):

- [Wie funktioniert Amazon GameLift Servers FlexMatch](#)
- [FlexMatch Schritte zur Integration](#)

Anpassungen der Platzierung von Spielsitzungen

Anpassungen an die Platzierung von Spielsitzungen geben dir die Möglichkeit, dein Platzierungssystem zu verfeinern, um deiner Spielerbasis das bestmögliche Spielerlebnis zu bieten. Du kannst die Platzierung im Hinblick auf Spielerkompatibilität und Präferenzen sowie

betriebliche Überlegungen wie Kosteneffizienz, geografische Verteilung und Ausfallsicherheit bei Betriebsunterbrechungen optimieren.

Passen Sie eine Warteschlange für Spielsitzungen an

In diesem Thema wird beschrieben, wie Sie Ihre Warteschlangen für Spielsitzungen anpassen können, um die bestmöglichen Entscheidungen über die Platzierung von Spielsitzungen zu treffen. Weitere Informationen zu Warteschlangen für Spielsitzungen und deren Funktionsweise finden Sie unter. [Platzierung der Spielsitzungen konfigurieren](#)

Für diese Amazon GameLift Servers Funktionen sind Warteschlangen erforderlich:

- [Matchmaking mit FlexMatch](#)
- [Eine Warteschlange für Spot-Instances erstellen](#)

Themen

- [Definieren Sie den Umfang einer Warteschlange](#)
- [Erstellen Sie eine Warteschlange mit mehreren Standorten](#)
- [Warteschlangenmetriken auswerten](#)

Definieren Sie den Umfang einer Warteschlange

In der Spielerpopulation deines Spiels gibt es möglicherweise Gruppen von Spielern, die nicht zusammen spielen sollten. Wenn du dein Spiel beispielsweise in zwei Sprachen veröffentlichst, sollte jede Sprache ihre eigenen Spieleserver haben.

Um die Platzierung von Spielsitzungen für deine Spielerpopulation einzurichten, erstelle für jedes Spielersegment eine separate Warteschlange. Grenzen Sie jede Warteschlange so ein, dass die Spieler auf den richtigen Spielservern platziert werden. Zu den gängigen Methoden zur Einteilung von Warteschlangen gehören:

- Nach geografischen Standorten. Wenn Sie Ihre Spieleserver in mehreren geografischen Gebieten einsetzen, können Sie Warteschlangen für Spieler an jedem Standort einrichten, um die Latenz der Spieler zu verringern.
- Durch Build- oder Skriptvarianten. Wenn du mehr als eine Variante deines Spielervers hast, unterstützt du möglicherweise Spielergruppen, die nicht an denselben Spielsitzungen teilnehmen können. Beispielsweise können Gameserver-Builds oder -Skripte verschiedene Sprachen oder Gerätetypen unterstützen.

- Nach Ereignistypen. Sie können eine spezielle Warteschlange erstellen, um Spiele für Teilnehmer an Turnieren oder anderen Sonderveranstaltungen zu verwalten.

Entwerfen Sie mehrere Warteschlangen

Abhängig von deinem Spiel und deinen Spielern möchtest du vielleicht mehr als eine Warteschlange für Spielsitzungen erstellen. Wenn dein Spiel-Client-Dienst eine neue Spielsitzung anfordert, gibt er an, welche Warteschlange für Spielsitzungen verwendet werden soll. Um dir bei der Entscheidung zu helfen, ob du mehrere Warteschlangen verwenden solltest, solltest du Folgendes beachten:

- Varianten deines Spieleservers. Du kannst für jede Variante deines Gameservers eine separate Warteschlange erstellen. Alle Flotten in einer Warteschlange müssen kompatible Spieleserver bereitstellen. Das liegt daran, dass Spieler, die die Warteschlange nutzen, um an Spielen teilzunehmen, in der Lage sein müssen, auf jedem Spieleserver der Warteschlange zu spielen.
- Verschiedene Spielergruppen. Sie können festlegen, wie Spielsitzungen je nach Spielergruppe Amazon GameLift Servers platziert werden. Beispielsweise benötigen Sie möglicherweise Warteschlangen, die auf bestimmte Spielmodi zugeschnitten sind und für die ein spezieller Instanztyp oder eine spezielle Laufzeitkonfiguration erforderlich ist. Oder vielleicht möchtest du eine spezielle Warteschlange einrichten, um Platzierungen für ein Turnier oder eine andere Veranstaltung zu verwalten.
- Statistiken zur Warteschlange von Spielsitzungen. Sie können Warteschlangen einrichten, je nachdem, wie Sie Metriken zur Platzierung von Spielsitzungen sammeln möchten. Weitere Informationen finden Sie unter [Amazon GameLift Servers-Metriken für Warteschlangen](#).

Erstellen Sie eine Warteschlange mit mehreren Standorten

Wir empfehlen ein Design mit mehreren Standorten für alle Warteschlangen. Dieses Design kann die Platzierungsgeschwindigkeit und die Stabilität des Hostings verbessern. Ein standortübergreifendes Design ist erforderlich, um anhand von Daten zur Spielerlatenz Spieler an Spielsitzungen mit minimaler Latenz teilnehmen zu lassen. Wenn Sie Warteschlangen mit mehreren Standorten erstellen, die Spot-Instance-Flotten verwenden, folgen Sie den Anweisungen unter [Senken Sie die Kosten für das Hosting von Spielen mit Spot-Flotten](#)

Eine Möglichkeit, eine Warteschlange mit mehreren Standorten zu erstellen, besteht darin, einer Warteschlange eine Flotte mit [mehreren Standorten](#) hinzuzufügen. Auf diese Weise kann die Warteschlange Spielsitzungen an beliebigen Standorten der Flotte platzieren. Aus Redundanzgründen kannst du auch andere Flotten mit unterschiedlichen Konfigurationen oder

Heimatorten hinzufügen. Wenn Sie eine Spot-Instance-Flotte mit mehreren Standorten verwenden, befolgen Sie die bewährten Methoden und fügen Sie eine On-Demand-Instance-Flotte mit denselben Standorten hinzu.

Das folgende Beispiel beschreibt den Prozess der Gestaltung einer einfachen Warteschlange mit mehreren Standorten. In diesem Beispiel verwenden wir zwei Flotten: eine Spot-Instance-Flotte und eine On-Demand-Instance-Flotte. Jede Flotte hat die folgenden AWS-Regionen Platzierungsorte: `us-east-1`, `us-east-2`, `ca-central-1`, und `us-west-2`.

Um eine einfache Warteschlange mit mehreren Standorten und Flotten mit mehreren Standorten zu erstellen

1. Wählen Sie einen Standort aus, an dem die Warteschlange erstellt werden soll. Sie können die Latenz bei Anfragen minimieren, indem Sie die Warteschlange an einem Ort platzieren, an dem Sie den Client-Service bereitgestellt haben. In diesem Beispiel erstellen wir die Warteschlange in `us-east-1`.
2. Erstellen Sie eine neue Warteschlange und fügen Sie Ihre Flotten mit mehreren Standorten als Warteschlangenziele hinzu. Die Reihenfolge der Ziele bestimmt, wie Spielsitzungen Amazon GameLift Servers platziert werden. In diesem Beispiel listen wir zuerst die Spot-Instance-Flotte und dann die On-Demand-Instance-Flotte auf.
3. Definieren Sie die Prioritätsreihenfolge der Spielsitzungen in der Warteschlange. Diese Reihenfolge bestimmt, wo die Warteschlange zuerst nach einem verfügbaren Spieleserver sucht. In diesem Beispiel verwenden wir die standardmäßige Prioritätsreihenfolge.
4. Definieren Sie die Reihenfolge der Standorte. Wenn Sie die Reihenfolge der Standorte nicht definieren, werden die Standorte in alphabetischer Reihenfolge Amazon GameLift Servers verwendet.

Game session placement locations

Locations where the queue can place new game sessions.

Locations

Choose locations

ca-central-1 ✕
Canada (Central)

us-west-2 ✕
US West (Oregon)

us-east-2 ✕
US East (Ohio)

us-east-1 ✕
US East (N. Virginia)

Destination order

An ordered list of fleets and aliases that the queue can use for game session placement.

	Region	Type	Name	
⋮	us-east-1 ▼	Fleet ▼	TestFleet-SPOT ▼	Remove
⋮	us-east-1 ▼	Fleet ▼	TestFleet-ONDEMAND ▼	Remove
Add Destination				

Game session placement priority

The values that GameLift uses to prioritize game session placement. The default order is latency, cost, destination, and location.

⋮

Latency

Prioritize locations with the lowest average player latency.

⋮

Cost

Prioritize destinations with the lowest current hosting cost.

⋮

Destination

Prioritize based on the defined destination order.

⋮

Location

Prioritize based on the defined location order.

▼ Location order

An ordered list of locations that the queue can use for game session placement.

Location

ca-central-1

▼

Remove

us-east-1

▼

Remove

us-east-2

▼

Remove

us-west-2

▼

Remove

Add locations

Warteschlangenmetriken auswerten

Verwenden Sie Metriken, um zu bewerten, wie gut Ihre Warteschlangen ausgeführt werden. Metriken zu Warteschlangen finden Sie in der [Amazon GameLift Servers Konsole](#) oder bei Amazon CloudWatch. Eine Liste und eine Beschreibung der Queue-Metriken finden Sie unter [Amazon GameLift Servers-Metriken für Warteschlangen](#).

Queue-Metriken können Aufschluss über Folgendes geben:

- **Gesamtleistung der Warteschlange** — Warteschlangenkennzahlen geben an, wie erfolgreich eine Warteschlange auf Platzierungsanfragen reagiert. Anhand dieser Kennzahlen können Sie auch feststellen, wann und warum Platzierungen fehlschlagen. Bei Warteschlangen mit manuell skalierten Flotten können die QueueDepth Kennzahlen AverageWaitTime und angeben, wann Sie die Kapazität für eine Warteschlange anpassen sollten.
- **FleetIQ Leistung des Algorithmus** — Für Platzierungsanfragen mit dem FleetIQ Algorithmus, Metriken zeigen, wie oft der Algorithmus die ideale Platzierung von Spielsitzungen findet. Bei der Platzierung kann der Einsatz von Ressourcen mit der geringsten Spielerlatenz oder Ressourcen mit den niedrigsten Kosten priorisiert werden. Es gibt auch Fehlermetriken, die die häufigsten Gründe dafür identifizieren Amazon GameLift Servers kann keine ideale Platzierung finden. Weitere Informationen zu den Metriken finden Sie unter [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#).
- **Standortspezifische Platzierungen** — Bei Warteschlangen mit mehreren Standorten zeigen Kennzahlen erfolgreiche Platzierungen nach Standort an. Für Warteschlangen, die den FleetIQ Diese Daten liefern nützliche Erkenntnisse darüber, wo Spieleraktivitäten stattfinden.

Bei der Auswertung von Kennzahlen für FleetIQ Beachten Sie die folgenden Tipps zur Leistung des Algorithmus:

- Um zu verfolgen, wie schnell die Warteschlange eine ideale Platzierung gefunden hat, verwenden Sie die PlacementsSucceeded Metrik in Kombination mit FleetIQ Kennzahlen für die niedrigste Latenz und den niedrigsten Preis.
- Um die Geschwindigkeit zu erhöhen, mit der eine Warteschlange eine ideale Platzierung findet, überprüfen Sie die folgenden Fehlermetriken:
 - Wenn der Wert hoch FirstChoiceOutOfCapacity ist, passen Sie die Kapazitätsskalierung für die Flotten der Warteschlange an.
 - Wenn die FirstChoiceNotViable Fehlermetrik hoch ist, schauen Sie sich Ihre Spot-Instance-Flotten an. Spot-Instance-Flotten gelten als nicht rentabel, wenn die Unterbrechungsrate für einen bestimmten Instance-Typ zu hoch ist. Um dieses Problem zu beheben, ändern Sie die Warteschlange so, dass Spot-Instance-Flotten mit unterschiedlichen Instance-Typen verwendet werden. Wir empfehlen, Spot-Instance-Flotten mit unterschiedlichen Instance-Typen an jedem Standort einzubeziehen.

Priorisieren Sie die Platzierung von Spielsitzungen

Amazon GameLift Servers verwendet einen Algorithmus, um zu bestimmen, wie die Ziele einer Warteschlange priorisiert werden und wo eine neue Spielsitzung stattfinden soll. Der Algorithmus basiert auf einem geordneten Satz von Kriterien. Sie können die standardmäßige Prioritätsreihenfolge verwenden oder die Reihenfolge anpassen. Sie können die Prioritätsreihenfolge einer Warteschlange jederzeit bearbeiten.

Standardmäßige Prioritätsreihenfolge

1. **Latenz** — Wenn die Anfrage zur Platzierung einer Spielsitzung standortspezifische Latenzdaten für Spieler enthält, wird die durchschnittliche Spielerlatenz an jedem Standort Amazon GameLift Servers berechnet und versucht, eine Spielsitzung an einem Flottenstandort mit dem niedrigsten Durchschnitt zu platzieren.
2. **Kosten** — Wenn eine Anfrage keine Latenzdaten enthält oder wenn mehrere Flotten dieselbe Latenz haben, werden die Hosting-Kosten Amazon GameLift Servers für jede Flotte berechnet. Die Hosting-Kosten einer Flotte variieren je nach Flottenart (Spot oder On-Demand), Instance-Typ und Standort.
3. **Ziel** — Wenn mehrere Flotten die gleiche Latenz und die gleichen Kosten haben, werden Flotten anhand der Zielreihenfolge, die in der Warteschlangenkonfiguration aufgeführt ist, Amazon GameLift Servers priorisiert.
4. **Standort** — Wenn bei Warteschlangen mit Flotten mit mehreren Standorten alle anderen Kriterien übereinstimmen, werden die Standorte der Flotte in alphabetischer Amazon GameLift Servers Reihenfolge priorisiert.

Passe an, wie eine Warteschlange die Platzierungen von Spielsitzungen priorisiert

Du kannst festlegen, wie eine Warteschlange die Platzierungskriterien priorisiert. Die Warteschlange wendet die benutzerdefinierte Priorisierung auf alle Anfragen zur Platzierung von Spielsitzungen an, die sie erhält.

Note

Wenn Sie eine benutzerdefinierte Prioritätskonfiguration erstellen und nicht alle vier Kriterien angeben, Amazon GameLift Servers werden alle fehlenden Kriterien automatisch in der Standardreihenfolge hinzugefügt.

Um die Prioritätskonfiguration einer Warteschlange anzupassen

Verwenden Sie die [Amazon GameLift ServersKonsole](#) oder die AWS Command Line Interface (AWS CLI), um eine benutzerdefinierte Prioritätskonfiguration zu erstellen.

Console

In der [Amazon GameLift ServersKonsole](#) können Sie die Prioritäten einer Warteschlange anpassen, wenn Sie eine neue Warteschlange erstellen oder eine bestehende Warteschlange aktualisieren. Wählen Sie eine AWS Region aus, in der Sie arbeiten möchten.

Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie Warteschlangen. Wählen Sie auf der Seite Warteschlangen eine bestehende Warteschlange aus und klicken Sie auf Bearbeiten.

1. Gehe zum Abschnitt Priorität der Platzierung von Spielsitzungen. Ziehen Sie jedes Prioritätskriterium per Drag-and-Drop, um die gewünschte Reihenfolge zu erstellen.
2. Gehen Sie zum Abschnitt Standortreihenfolge. Fügen Sie alle Standorte hinzu, die Sie priorisieren möchten. Diese Liste ist nützlich, wenn die Warteschlange Flotten mit mehreren Standorten enthält. Sie müssen mindestens einen Standort angeben. Die Standorte, die Sie hier angeben, werden zuerst priorisiert, gefolgt von allen anderen Standorten in den Zielen der Warteschlange.
3. Wählen Sie Änderungen speichern aus.

AWS CLI

Verwenden Sie den [update-game-session-queue](#)Befehl mit der `--priority-configuration` Option, um die Prioritätsreihenfolge einer Warteschlange anzupassen. Amazon GameLift Serversaktualisiert eine Warteschlange in Ihrer aktuellen AWS Standardregion, oder Sie können ein `--region` Tag hinzufügen, um eine andere AWS Region anzugeben.

Die folgende Beispielanforderung fügt die Prioritätskonfiguration für eine angegebene Warteschlange hinzu oder aktualisiert sie

```
aws gamelift update-game-session-queue \  
  --name "example-queue-with-priority" \  
  --priority-configuration \  
    PriorityOrder="COST", "LOCATION", "DESTINATION", LocationOrder="us-east-1", "us-  
east-2", "ca-central-1", "us-west-2" \  
  --region us-east-1
```

Priorisieren Sie Platzierungen nach der Latenz der Spieler

Wenn du deinen Spielern das bestmögliche Spielerlebnis bieten und eine minimale Latenz sicherstellen möchtest, führe bei der Einrichtung deines Platzierungssystems für Spielsitzungen die folgenden Schritte durch:

- Richten Sie Ihre Warteschlange so ein, dass bei der Auswahl, wo Spielsitzungen stattfinden sollen, der Latenz Priorität eingeräumt wird. Die Latenz steht standardmäßig ganz oben auf der Prioritätenliste. Sie können auch die Prioritätskonfiguration Ihrer Warteschlange anpassen und festlegen, wo die Latenz in der Prioritätsreihenfolge angeordnet werden soll.
- Richten Sie Richtlinien für die Spielerlatenz für Ihre Warteschlange ein. Mithilfe von Latenzrichtlinien kannst du feste Grenzen für die Latenz festlegen, die bei der Platzierung einer Spielsitzung zulässig sein soll. Wenn eine Spielsitzung nicht stattfinden Amazon GameLift Servers kann, ohne die Limits zu überschreiten, wird die Platzierungsanfrage nach einer Zeitüberschreitung unterbrochen und schlägt fehl. Sie können eine einzelne Latenzrichtlinie einrichten, oder Sie können eine Reihe von Richtlinien erstellen, die das Latenzlimit im Laufe der Zeit schrittweise lockern. Mit einer Reihe von Richtlinien können Sie anfänglich sehr niedrige Latenzgrenzen festlegen und Spieler mit höheren Latenzen auch nach einer kurzen Verzögerung berücksichtigen. Einzelheiten zur Erstellung von Latenzrichtlinien finden Sie unter [Erstellen Sie eine Latenzrichtlinie für Spieler](#).
- Wenn Sie Anfragen zur Platzierung von Spielsitzungen stellen (siehe [StartGameSessionPlacement](#)), geben Sie die Latenzdaten für jeden Spieler an. Die Daten zur Spielerlatenz beinhalten einen Wert für jeden möglichen Ort, an dem eine Spielsitzung stattfinden könnte. Für eine Warteschlange, die Spielsitzungen in AWS-Regionen us-east-2 und ca-central-1 platziert, könnten die Latenzdaten beispielsweise wie folgt aussehen:

```
"PlayerLatencies": [  
  { "LatencyInMilliseconds": 100, "PlayerId": "player1", "RegionIdentifier": "us-east-2" },  
  { "LatencyInMilliseconds": 100, "PlayerId": "player1", "RegionIdentifier": "ca-central-1" },  
  { "LatencyInMilliseconds": 150, "PlayerId": "player2", "RegionIdentifier": "us-east-2" },  
  { "LatencyInMilliseconds": 150, "PlayerId": "player2", "RegionIdentifier": "ca-central-1" }  
]
```

Verwenden Amazon GameLift Servers Sie die UDP-Ping-Beacons, um genaue Latenzmessungen zu erhalten. Mit diesen Endpunkten können Sie die tatsächliche UDP-Netzwerklatenz zwischen den Player-Geräten und jedem der potenziellen Hosting-Standorte messen, was zu genaueren Platzierungsentscheidungen führt als mit ICMP-Pings. Weitere Informationen zur Verwendung von UDP-Ping-Beacons zur Messung der Latenz finden Sie unter. [UDP-Ping-Beacons](#)

Priorisieren Sie Platzierungen nach Standort

Sie können eine Warteschlange so konfigurieren, dass Platzierungen für Spielsitzungen auf der Grundlage einer priorisierten Liste geografischer Standorte vorgenommen werden. Der Standort ist eines der Kriterien, die bestimmen, wie eine Warteschlange auswählt, wo eine neue Spielsitzung stattfinden soll. Standardmäßig wird der Standort an vierter Stelle nach Latenz, Kosten und Ziel priorisiert.

Bei der Platzierung von Spielsitzungen haben Ziel und Ort etwas unterschiedliche Bedeutungen:

- Ziel bezieht sich auf eine bestimmte Flotte und umfasst alle Hosting-Ressourcen der Flotte, unabhängig davon, wo sie eingesetzt werden. Bei der Priorisierung nach Ziel Amazon GameLift Servers kann eine Platzierung an einem beliebigen Standort in der Flotte vorgenommen werden. Verwaltete Flotten mit mehreren Standorten und Anywhere-Flotten können über Hosting-Ressourcen verfügen, die an einem oder mehreren Standorten bereitgestellt werden.
- Standort bezieht sich auf eine bestimmte geografische Position, an der die Hosting-Ressourcen einer Flotte eingesetzt werden. Eine Flotte kann mehrere Standorte haben, zu denen Local Zones oder benutzerdefinierte Standorte (für eine Anywhere-Flotte) gehören AWS-Regionen können. Eine an einem einzigen Standort verwaltete Flotte hat einen Standort und es handelt sich immer um einen AWS-Region. Eine an mehreren Standorten verwaltete Flotte hat eine Heimatregion und kann auch abgelegene Standorte haben. Eine Anywhere-Flotte hat einen oder mehrere benutzerdefinierte Standorte.

Sucht bei der Priorisierung von Platzierungen nach Standort nach allen Warteschlangenzielen, Amazon GameLift Servers die den bevorzugten Standort enthalten, und sucht dort nach einer verfügbaren Hosting-Ressource. Wenn es mehrere Ziele mit dem bevorzugten Standort gibt, Amazon GameLift Servers wird mit den Kriterien für die nächste Priorität (Kosten, Latenz, Ziel) fortgefahren.

Es gibt mehrere Möglichkeiten, wie Sie beeinflussen können, wie die Standorte einer Warteschlange priorisiert werden

- Konfigurieren Sie, wie die Warteschlange alle Anfragen zur Platzierung von Spielsitzungen behandelt:
 - Fügen Sie der Warteschlange eine Prioritätskonfiguration hinzu. Die Prioritätskonfiguration einer Warteschlange umfasst eine geordnete Liste von Standorten. Sie können einen oder mehrere Standorte angeben, die priorisiert werden sollen. Diese Liste schließt keine Standorte aus, sie gibt lediglich an, Amazon GameLift Servers wo zuerst nach einer verfügbaren Hosting-Ressource gesucht werden muss. Eine geordnete Standortliste wird häufig verwendet, wenn Sie den Großteil des Datenverkehrs an einen oder mehrere bestimmte geografische Standorte leiten und zusätzliche Standorte als Backup-Kapazität verwenden möchten. Fügen Sie eine Prioritätskonfiguration hinzu, indem Sie anrufen [UpdateGameSessionQueue](#).
 - Fügen Sie der Warteschlange eine Filterkonfiguration hinzu. Eine Filterkonfiguration ist eine Zulassungsliste für die Warteschlange. Sie weist darauf Amazon GameLift Servers hin, dass bei der Suche nach einer verfügbaren Hosting-Ressource alle Standorte ignoriert werden sollen, die nicht auf der Liste stehen. Es gibt zwei gängige Verwendungszwecke für eine Filterkonfiguration. Erstens können Sie bei Flotten mit mehreren Standorten einen Filter verwenden, um einige Standorte der Flotte auszuschließen. Zweitens möchten Sie möglicherweise vorübergehend Platzierungen an einem bestimmten Standort verbieten. Beispielsweise könnten an einem Standort vorübergehende Probleme auftreten. Da Sie die Filterkonfiguration einer Warteschlange jederzeit aktualisieren können, können Sie Standorte ganz einfach nach Bedarf hinzufügen oder entfernen. Fügen Sie eine Filterkonfiguration hinzu, indem Sie anrufen [UpdateGameSessionQueue](#).
- Verwenden Sie spezielle Anweisungen für individuelle Platzierungsanfragen:
 - Fügen Sie einer Platzierungsanfrage für eine Spielsitzung eine Liste mit überschriebenen Prioritäten bei. Sie können bei jeder [StartGameSessionPlacement](#)Anfrage eine alternative Prioritätsliste mit Standorten bereitstellen. Diese Liste ersetzt effektiv die konfigurierte Priorisierung der Warteschlange für Standorte nur für diese eine Anfrage. Sie hat keine Auswirkungen auf andere Anfragen. Für diese Override-Funktion gelten einige Anforderungen:
 - Verwenden Sie eine Override-Liste nur für eine Warteschlange, für die eine Prioritätskonfiguration mit LOCATION erster Priorität eingerichtet wurde.
 - Nehmen Sie in derselben Platzierungsanfrage keine Daten zur Spielerlatenz auf. Die Einbeziehung von Latenzdaten führt zu Konflikten bei der Priorisierung von Standorten, die nicht gelöst Amazon GameLift Servers werden können.
 - Entscheiden Sie, wie Sie vorgehen Amazon GameLift Servers möchten, wenn eine verfügbare Ressource auf der Prioritätsliste nicht gefunden werden kann. Wählen Sie, ob Sie auf die anderen Standorte der Warteschlange zurückgreifen oder Platzierungen auf die

Überschreibungsliste beschränken möchten. Standardmäßig wird versucht, Amazon GameLift Servers die Platzierung an den anderen Orten der Warteschlange zu versuchen.

- Aktualisieren Sie die Filterkonfiguration der Warteschlange nach Bedarf, indem Sie beispielsweise Standorte zur Override-Liste hinzufügen. Die Überschreibungsliste macht die Filterliste nicht ungültig.

Erstellen Sie eine Latenzrichtlinie für Spieler

Wenn Ihre Platzierungsanfragen Daten zur Spielerlatenz enthalten, werden Amazon GameLift Servers, werden Spielsitzungen an Orten mit der niedrigsten durchschnittlichen Latenz für alle Spieler gefunden. Wenn Sie Spielsitzungen auf der Grundlage der durchschnittlichen Spielerlatenz platzieren, wird Amazon GameLift Servers verhindert, dass die meisten Spieler an Spielen mit hoher Latenz teilnehmen. Allerdings werden Spieler Amazon GameLift Servers immer noch mit extremer Latenz platziert. Um diesen Spielern gerecht zu werden, sollten Sie Richtlinien zur Spielerlatenz erstellen.

Eine Richtlinie zur Spielerlatenz Amazon GameLift Servers verhindert, dass eine angeforderte Spielsitzung an einer Stelle platziert wird, an der bei Spielern in der Anfrage eine Latenz über dem Maximalwert auftreten würde. Richtlinien zur Spielerlatenz können auch Amazon GameLift Servers verhindern, dass Anfragen für Spielsitzungen Spielern mit höherer Latenz zugeordnet werden.

Tip

Um latenzspezifische Regeln zu verwalten, z. B. eine ähnliche Latenz für alle Spieler in einer Gruppe vorzuschreiben, kannst du [Amazon GameLift Servers FlexMatch](#) latenzbasierte Matchmaking-Regeln erstellen.

Stellen Sie sich zum Beispiel diese Warteschlange mit einem Timeout von 5 Minuten und den folgenden Latenzrichtlinien für Spieler vor:

1. Verbringen Sie 120 Sekunden damit, nach einem Ort zu suchen, an dem alle Spielerlatenzen weniger als 50 Millisekunden betragen.
2. Verbringen Sie 120 Sekunden damit, nach einem Ort zu suchen, an dem alle Spielerlatenzen weniger als 100 Millisekunden betragen.
3. Verbringen Sie die verbleibende Wartezeit bis zum Timeout damit, nach einem Ort zu suchen, an dem die Latenzen aller Spieler weniger als 200 Millisekunden betragen.

Create queue

Queue settings

Name

The name must be unique and have 1-128 characters. Valid characters: A-Z, a-z, 0-9, and - (hyphen).

Timeout

Specify how long GameLift tries to place a game session before stopping.

 seconds

Must be 10-600 seconds.

 We recommend setting player latency policies, unless you're using GameLift FlexMatch.



Player latency policies - *optional*

Add policies to help place players into games with lower latency. Use multiple policies to reduce latency requirements per policy so that each player eventually finds a match.

0 seconds left to allocate 100%

Period start

Seconds

Period end

Seconds

Max player latency

Milliseconds

Remove

Seconds

Seconds

Milliseconds

Remove

Seconds

Seconds

Milliseconds

Remove

Add policy

Eine Warteschlange für Spot-Instances erstellen

Durch den Einsatz von Spot-Flotten können Sie potenziell erhebliche Einsparungen bei den Hosting-Kosten erzielen. Weitere Informationen zu Spot-Flotten und deren Verwendung finden Sie unter. [On-Demand-Instances im Vergleich zu Spot Instances](#)

Wenn deine Game-Hosting-Lösung Spot-Flotten beinhaltet, musst du eine Warteschlange für die Platzierung von Spielsitzungen verwenden. Amazon GameLift Servers verwendet Warteschlangen, um mehrere Ressourcen zum Hosten von Spielen zu durchsuchen und die beste verfügbare Ressource für die Ausrichtung einer neuen Spielsitzung auszuwählen. Bei Spot-Flotten sind Warteschlangen besonders wichtig, um die Hosting-Kosten zu minimieren und mögliche Spot-Unterbrechungen zu vermeiden. Dieses Thema hilft Ihnen dabei, eine stabile Warteschlange einzurichten, in der auch bei Unterbrechungen, Verlangsamungen und Ausfällen weiterhin Spiele für Spieler gehostet werden können. Sie können anhand verschiedener Faktoren, einschließlich der Hosting-Kosten, anpassen, wie die Warteschlange die verfügbaren Hosting-Ressourcen priorisiert.

Nutzen Sie FlexMatch für das Matchmaking? Du kannst eine Warteschlange mit Spot-Flotten verwenden, um Spielsitzungen für deine Spiele zu platzieren.

Implementierungsaufgaben für Spot-Flotten

Wenn du deine Game-Hosting-Lösung für die Verwendung von Spot-Flotten erstellst oder aktualisierst, führe die folgenden Aufgaben aus. Ausführlichere Anleitungen zum Erstellen einer Warteschlange, die die Verfügbarkeit und Stabilität von Spot optimiert, finden Sie unter [Senken Sie die Kosten für das Hosting von Spielen mit Spot-Flotten](#)

1. Wählen und erstellen Sie eine Reihe von Flottenzielen für Ihre Warteschlange für Spielsitzungen.

Entscheiden Sie zunächst, wo Ihre Warteschlange Spielsitzungen platzieren soll. Eine Warteschlange kann mehrere Flotten durchsuchen, um die bestmögliche Platzierung zu finden. Jede Flotte hat einen Instance-Typ, kann aber auch mehrere geografische Standorte haben. Warteschlangen mit Flotten, die sowohl hinsichtlich des Standorts als auch des Instance-Typs Abwechslung bieten, führen eher zu erfolgreichen Platzierungen. Sehen Sie sich diese bewährten Methoden zum Entwerfen einer effektiven und belastbaren, SPOT-optimierten Warteschlange an.

2. Erstellen Sie Ihre SPOT-optimierte Warteschlange für Spielsitzungen.

Erstelle eine Warteschlange und konfiguriere sie für deine Spot-Flotten. Weitere Informationen zum Erstellen und Konfigurieren der neuen Warteschlange finden Sie unter [Eine Warteschlange für Spielsitzungen erstellen](#). Sie können die Amazon GameLift Servers Konsole oder die AWS CLI verwenden, um eine Warteschlange zu erstellen oder zu bearbeiten.

- Fügen Sie die Flottenziele aus Schritt 1 hinzu.

- Priorisieren Sie die Zielreihenfolge nach Bedarf. Standardmäßig werden die Kosten vor dem Ziel Amazon GameLift Servers priorisiert, sodass die Zielreihenfolge nur verwendet wird, wenn die niedrigsten Kosten zwischen den Zielen identisch sind.
 - Wenn Sie die Kosten für das Hosting von Spielen vor der Latenz der Spieler priorisieren möchten, geben Sie eine benutzerdefinierte Platzierungspriorität an. Siehe [Priorisieren Sie die Platzierung von Spielsitzungen](#).
3. Aktualisieren Sie andere Komponenten in Ihrer Lösung, um die neue Warteschlange zu verwenden.

Wenn Ihre Lösung eine SPOT-optimierte Warteschlange verwendet, um neue Spielsitzungen zu starten, verhindert die Warteschlange automatisch, dass Spielsitzungen mit Flotten platziert werden, bei denen die Wahrscheinlichkeit einer Unterbrechung hoch ist. Stattdessen durchsucht sie alle brauchbaren Flotten nach Ressourcen, die deinen definierten Prioritäten entsprechen, einschließlich Spielerlatenz, Hosting-Kosten und Zielreihenfolge.

- Falls du es nicht verwendest FlexMatch — Aktualisiere deinen Backend-Service, um die neue Spot-optimierte Warteschlange für Spielsitzungen anzugeben. Der Backend-Service stellt API-Anfragen im Namen eures Spielclients (using `startGameSessionPlacement()`) Amazon GameLift Servers an, und für jede Anfrage muss ein Warteschlangenname angegeben werden. Hilfe bei der Implementierung der Platzierungen von Spielsitzungen in Ihrem Spielclient finden Sie unter [Spielsitzungen erstellen](#).
 - Falls du verwendest FlexMatch — Aktualisiere deine Matchmaking-Konfiguration, um Anfragen für Spielsitzungen an die neue SPOT-optimierte Warteschlange zu senden. Wenn das Matchmaking-System ein Spielermatch bildet, sendet es eine Anfrage zur Platzierung einer Spielsitzung an die dafür vorgesehene Warteschlange, um eine neue Spielsitzung für das Spiel zu starten. Nur bei Matchmaking-Konfigurationen, bei denen der FlexMatch Modus auf „Verwaltet“ eingestellt ist, kann eine Platzierungswarteschlange festgelegt werden. Sie können eine Matchmaking-Konfiguration über die AWS CLI oder die Amazon GameLift Servers Konsole aktualisieren (siehe [Matchmaking-Konfiguration bearbeiten](#)).
4. Überprüfen Sie die Leistung Ihrer Spot-Flotten und Warteschlangen.

Sehen Sie sich Amazon GameLift Servers Metriken in der Amazon GameLift Servers Konsole oder bei Amazon CloudWatch an, um die Leistung zu überprüfen. Weitere Informationen zu den Amazon GameLift Servers-Metriken finden Sie unter [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#). Zu den wichtigsten Metriken gehören:

- **Unterbrechungsrate** — Verwenden Sie die `GameSessionInterruptions` Messwerte `InstanceInterruptions` und, um die Anzahl und Häufigkeit von SPOT-bedingten Unterbrechungen für Instances und Spielsitzungen nachzuverfolgen. Spielsitzungen auf zurückgeforderten Instanzen haben den Status `TERMINATED` und den Statusgrund von `INTERRUPTED`.
- **Effektivität der Warteschlangen** — Verfolge die Erfolgsquoten bei der Platzierung, die durchschnittliche Wartezeit und die Warteschlangentiefe, um sicherzugehen, dass Spot-Flotten deine Leistung in der Warteschlange nicht beeinträchtigen.
- **Flottennutzung** — Überwachen Sie Daten zu Instanzen, Spiel- und Spielersitzungen. Die Nutzung Ihrer On-Demand-Flotten kann ein Indikator dafür sein, dass Warteschlangen dazu führen, dass Sie nicht in Ihre Spot-Flotten aufgenommen werden, um Störungen zu vermeiden.

Bewährte Methoden für Warteschlangen mit Spot-Flotten

Verwenden Sie beim Erstellen von Flotten und Warteschlangen für Spot-Instances die folgenden bewährten Methoden.

- **Erweitern Sie die geografische Abdeckung Ihrer Warteschlange.** Auch wenn sich deine Spieler in einem einzigen Cluster befinden AWS-Region, füge deiner Spot-Flotte benachbarte Standorte hinzu. Dieser Ansatz verbessert die Fähigkeit der Warteschlange, die Kapazität bei regionalen Verlangsamungen, Ausfällen und Spot-Unterbrechungen aufrechtzuerhalten. Flotten mit mehreren Standorten funktionieren sowohl mit Spot- als auch mit On-Demand-Instances.
- **Diversifizieren Sie die Abdeckung der Instance-Typen Ihrer Warteschlange.** Amazon GameLift Servers bewertet die Rentabilität von Spot anhand des Instance-Typs, sodass Spot-Flotten mit einer Vielzahl von Instance-Typen die Wahrscheinlichkeit verringern, dass mehrere Spot-Flotten gleichzeitig nicht rentabel sind. Schließt mindestens zwei Spot-Flotten mit unterschiedlichen Instance-Typen an jedem Standort ein.

Note

Die Preise basieren auf den Instances, die Sie verwenden, nicht auf der Anzahl der Flotten. Der Betrieb von fünf Flotten mit jeweils 10 Instances entspricht dem Betrieb einer Flotte mit 50 Instances zu ähnlichen Kosten. Die Preise variieren je nach Instance-Typ, Größe und Standort.

Tipps zur Gruppierung von Spot-Instance-Typen:

- Verwenden Sie Instance-Typen derselben Familie, wie `m6g.medium`, `m6g.large`, und `m6g.xlarge`. Größere Instanztypen kosten mehr, können aber auch mehrere Spielsitzungen gleichzeitig hosten.
- Wählen Sie allgemein verfügbare Instanztypen aus. Typischerweise bieten Familien älterer Generationen (wie C5, M5 und R5) und gängige Größen (wie `.large`, `.xlarge` und `.2xlarge`) eine bessere Verfügbarkeit.
- Sehen Sie sich in der Konsole die Preisentwicklung von 30 bis 90 Tagen an. Amazon GameLift Servers Suchen Sie nach Instance-Typen mit konsistenten Verfügbarkeitsmustern.
- Verwenden Sie die Amazon GameLift Servers Konsole, das Tool zur Flottenerstellung, um die Standortabdeckung für Instance-Typen zu untersuchen.
- Fügen Sie On-Demand-Flotten für Backup-Kapazität hinzu. Spiele-Hosting kann zu On-Demand-Flotten wechseln, wenn Spot-Flotten nicht verfügbar sind. Stell an jedem Standort mindestens eine On-Demand-Flotte auf, um die Latenz der Spieler gering zu halten. Fügen Sie auto-scaling zu Ihren On-Demand-Backup-Flotten hinzu, sodass Sie sie herunterskalieren können, bis sie benötigt werden.
- Weisen Sie allen Flottenzielen Aliase zu. Erstellen Sie Aliase für jedes Ziel Ihrer Warteschlange. Aliase machen es einfacher und effizienter, wann immer Sie Flotten austauschen müssen.
- Wenden Sie eine Strategie zur Priorisierung von Warteschlangen an. Du kannst anpassen, wie in einer Warteschlange Prioritäten gesetzt werden, wo Spielsitzungen platziert werden sollen (weitere Informationen findest du unter [Priorisieren Sie die Platzierung von Spielsitzungen](#)). Bei Spot-optimierten Warteschlangen wird durch die Priorisierung nach Kosten sichergestellt, dass wann immer möglich kostengünstige Spot-Flotten eingesetzt werden.

Sie können bestimmte Flotten auch priorisieren, indem Sie eine Zielreihenfolge angeben.

Beispielsweise weisen einige Benutzer eine Reihe primärer Flotten für den regelmäßigen Gebrauch und auch eine Reihe von sekundären Flotten als Ersatzflotten zu. In diesem Szenario legen Sie die Zielreihenfolge der Warteschlange so fest, dass die primären Flotten zuerst aufgeführt werden. Konfigurieren Sie dann die Prioritätsreihenfolge der Warteschlange mit dem Ziel, gefolgt von den Kosten.

Anpassungen der Hosting-Ressourcen

Dieser Abschnitt bietet erweiterte Optionen für die Konfiguration und Verwaltung Ihrer Amazon GameLift Servers Infrastruktur, um spezifische Leistungs-, Kosten- und Betriebsanforderungen zu erfüllen. In den Themen in diesem Abschnitt wird insbesondere beschrieben, wie Sie Ihre Amazon GameLift Servers verwalteten Hosting-Ressourcen so anpassen können, dass sie am besten zu Ihrem Spiel und Ihren Spielern passen.

Einige der Entscheidungen, die du in Betracht ziehen solltest:

- Wo können Sie Hosting-Ressourcen für Ihre Spieler einsetzen? Die Latenz im Gameplay ist ein wichtiger Faktor bei der Auswahl der geografischen Standorte deiner Flotte, aber es gibt auch andere Faktoren, die je nach Standort variieren, darunter die Verfügbarkeit der Ressourcen und die Kosten.
- Welche EC2 Instance-Typen werden dein Spiel am besten unterstützen? Wählen Sie aus den verfügbaren Instance-Typen, die die beste Kombination aus Rechenarchitektur, Arbeitsspeicher, Speicher und Netzwerkkapazität nutzen.
- Welche Größe des Instance-Typs benötigen Sie? Wählen Sie die Größe des Instance-Typs auf der Grundlage der Ressourcenanforderungen (Speicher und CPU) Ihrer Gameserver-Software und anderen Faktoren aus.
- Sollte Ihre Flotte On-Demand-Instances oder Spot-Instances verwenden? Überlegen Sie, ob Sie von niedrigeren Spot-Preisen profitieren können und ob die Wahrscheinlichkeit von Spot-Unterbrechungen Ihrer Spielsitzungen Amazon GameLift Servers ausreichend verringert wird.
- Wie soll Ihre Gameserver-Software auf jeder Flotteninstanz ausgeführt werden? Die Laufzeitkonfiguration legt fest, welche Serversoftware wie ausgeführt werden soll.
- Funktioniert die Standardkonfiguration für Containerflotten für dein Spiel? Amazon GameLift Servers nimmt Ihnen viel Arbeit bei der Optimierung Ihrer Containerflottenkonfigurationen ab, aber Sie können die meisten Konfigurationseinstellungen anpassen.

Wählen Sie Rechenressourcen für eine verwaltete Flotte

Für Amazon GameLift Servers verwaltetes Hosting, einschließlich verwalteter EC2- und verwalteter Container, stellt der Dienst Ihre Spieleserver auf Flotten von Rechenressourcen in der bereit. AWS Cloud Wenn Sie eine verwaltete Flotte erstellen, möchten Sie die Hosting-Ressourcen so

konfigurieren, dass sie am besten zu Ihrem Spiel passen. In diesem Thema werden wichtige Entscheidungspunkte bei der Auswahl und Konfiguration deiner Spiele-Hosting-Flotten behandelt.

Note

Wenn Sie eine Hybridlösung mit Anywhere und Amazon GameLift Servers verwalteten Flotten entwickeln möchten, können Sie anhand dieser Themen verwaltete Flotten als Ergänzung zu Ihren eigenen, selbst verwalteten Ressourcen entwerfen. Siehe [Stellen Sie Hosting-Flotten bereit für Amazon GameLift Servers](#).

Themen

- [Geografische Standorte](#)
- [Betriebssysteme](#)
- [Instance-Typen](#)
- [On-Demand-Instances im Vergleich zu Spot Instances](#)
- [Servicekontingente](#)

Geografische Standorte

Überlegen Sie, wo Sie Ihre Spieleserver einsetzen möchten. Im Allgemeinen möchten Sie Ihre Spieleserver so nah wie möglich an Ihren Spielern platzieren, um das bestmögliche Spielerlebnis zu bieten. Für Amazon GameLift Servers verwaltetes Hosting kannst du dich dafür entscheiden, Spieleserver in einer der unterstützten AWS-Regionen und Local Zones zu platzieren. Wenn Sie eine Hybridlösung entwickeln, sollten Sie sich überlegen, wie der Einsatz einer verwalteten Flotte die Standorte Ihrer selbstverwalteten Amazon GameLift Servers Anywhere-Flotten ergänzen kann.

Für die meisten Entwicklungs- und Testszenarien ist die Bereitstellung an einem einzigen Standort sinnvoll. Bei der Vorbereitung auf die Markteinführung und darüber hinaus gibt es viele Gründe, die für eine Bereitstellung an mehreren geografischen Standorten sprechen. Dazu gehören die Unterstützung einer großen Gruppe von Spielern und die Verbesserung der allgemeinen Widerstandsfähigkeit und Zuverlässigkeit des Spielhostings. Mehrere Standorte können auch das Spielerlebnis verbessern, indem sie die Platzierung von Spielsitzungen beschleunigen und mehr Auswahlmöglichkeiten bei der Optimierung von Platzierungen im Hinblick auf Latenz und Kosten bieten.

Eine Liste der Standorte, die von unterstützt werden, Amazon GameLift Servers und weitere Informationen zu Standorten für alle Flottenarten finden Sie unter. [Amazon GameLift Servers Servicestandorte](#)

Flotten mit mehreren Standorten

Eine einzige verwaltete Flotte kann Ressourcen an mehreren Standorten einsetzen. Sie können die Kapazität für jeden einzelnen Standort in einer Flotte mit mehreren Standorten manuell festlegen.

Vorteile der Verwendung einer Flotte mit mehreren Standorten:

- Vereinfachte Flottenbereitstellung und -verwaltung — Sie stellen die Gameserver-Software und die Flottenkonfiguration Amazon GameLift Servers bereit und stellen sie auf Flotteninstanzen an mehreren Standorten bereit (einmal erstellen, überall bereitstellen). In einer Produktionsflotte kannst du alle Standorte in einer Flotte einsehen und verwalten, anstatt mehrere Flotten verwalten zu müssen, die sich jeweils in einer anderen Region befinden.
- Verfügbarkeit lokaler Zonen — Wenn Sie eine lokale Zone verwenden möchten, müssen Sie eine Flotte mit mehreren Standorten mit einem AWS-Region Heimatstandort und Local Zones als entfernten Standorten einrichten. Local Zones sind Erweiterungen davon AWS-Regionen , die für Bereiche und Kunden, die sie benötigen, eine noch geringere Latenz bieten können. Sie können jeder Flotte mit mehreren Standorten eine lokale Zone hinzufügen. Sie müssen die der lokalen Zone übergeordnete AWS-Region Zone nicht hinzufügen.
- Kompatibilität mit Warteschlangen für Spielsitzungen — Sie können Warteschlangen für die Platzierung von Spielsitzungen mit einer oder mehreren Flotten mit mehreren Standorten erstellen. Dieser Ansatz gibt der Warteschlange Flexibilität bei der Priorisierung und Auswahl von Orten für die Ausrichtung einer neuen Spielsitzung.
- Effiziente Ressourcennutzung — Wenn die auto-scaling aktiviert ist, Amazon GameLift Servers kann die Kapazitätsskalierung an allen Standorten in einer Flotte besser optimiert werden.

Tipps für den Einsatz von Flotten mit mehreren Standorten:

- Suchen Sie nach Kontingenten für die Anzahl der Standorte pro Flotte AWS-Region . Siehe [Amazon GameLift Servers-Servicekontingente](#).
- Nicht alle Instance-Typen sind an allen Standorten verfügbar. Abhängig von den von Ihnen ausgewählten Standorten stehen Ihnen möglicherweise eingeschränkte Optionen für Instanztypen zur Verfügung. Die Amazon GameLift Servers Konsole bietet nützliche Tools, die Ihnen helfen, das richtige Gleichgewicht zwischen Standorten und Instance-Typen zu finden.

- Erwägen Sie [UDP-Ping-Beacons](#) die Verwendung zur Erfassung von Spielerlatenzdaten für all Ihre Flottenstandorte. Amazon GameLift Servers kann diese Daten verwenden, um Spielsitzungen für eine geringe Latenz zu optimieren und zu verhindern, dass Spieler an Sitzungen mit inakzeptabel hoher Latenz teilnehmen. Diese speziellen Endpunkte akzeptieren UDP-Nachrichten anstelle von herkömmlichen ICMP-Pings und bieten präzise Latenzmessungen, die Sie bei der Auswahl der optimalen Flottenstandorte unterstützen.

Betriebssysteme

Alle Instances in einer verwalteten Flotte werden mit einem Amazon Machine Image (AMI) bereitgestellt, das eine vollständige Laufzeitumgebung für Ihre Gameserver-Software bietet. Bei verwalteten EC2-Flotten geben Sie das Betriebssystem des Spielserver-Builds an, wenn Sie den Build auf hochladen. Amazon GameLift Servers Für verwaltete Container-Flotten geben Sie das Betriebssystem in der Container-Gruppendefinition an. Weitere Informationen zu den neuesten AMI-Versionen finden Sie unter [Amazon GameLift Servers AMI-Versionen](#).

AMI-Versionen werden regelmäßig aktualisiert. Wenn Sie eine neue Flotte erstellen, Amazon GameLift Servers weist Ihnen die neueste verfügbare Version des AMI zu, das Sie für Ihren Spiel-Build ausgewählt haben. Alle Instances, die in dieser Flotte bereitgestellt werden, verwenden dieselbe Version. Um Ihre AMI-Version mit den neuesten Sicherheits- und Softwareupdates auf dem neuesten Stand zu halten, müssen Sie Ihre Flotten regelmäßig austauschen. Als bewährte Methode empfehlen wir, Ihre verwalteten Flotten alle 30 Tage auszutauschen, um die Laufzeitumgebung für Ihre Spieleserver aufrechtzuerhalten. Anleitungen finden Sie unter [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#).

Instance-Typen

Der Instanztyp einer verwalteten Flotte bestimmt die Art der Hardware, die für alle Flotteninstanzen eingesetzt wird. Instance-Typen sind in der Regel in verschiedenen Größen erhältlich. Alle Amazon GameLift Servers verwalteten Flotten verwenden Amazon EC2 EC2-Instances und unterstützen eine Vielzahl von Instance-Typen, die unterschiedliche Kombinationen von Rechenleistung, Arbeitsspeicher, Speicher und Netzwerkfunktionen bieten. Die Verfügbarkeit von Instance-Typen hängt von den ausgewählten Standorten ab.

Die Amazon GameLift Servers Konsole bietet nützliche Tools, die dir dabei helfen, die richtigen Instanztypen für deinen Spiel-Build und deine Einsatzorte zu finden. Für verwaltete Containerflotten bietet die Konsole auch Hinweise zur CPU-Leistung und den Speicheranforderungen Ihres Spiels.

Wenn du einen der verfügbaren Instance-Typen für dein Spiel auswählst, solltest du Folgendes berücksichtigen:

- Die Rechenarchitektur deines Spieleservers: x64 oder Arm (AWS Graviton).

 Note

Graviton Arm-Instances erfordern einen Server-Build für ein Linux-AMI. Server-SDK 5.1.1 oder neuer ist für C++ und C# erforderlich. Server-SDK 5.0 oder neuer ist für Go erforderlich. Diese Instances bieten keine out-of-the-box Unterstützung für die Mono-Installation auf Amazon Linux 2023 (AL2023) oder Amazon Linux 2 (AL2).

- Die Rechen-, Speicher- und Speicheranforderungen Ihres Gameserver-Builds.
- Die Größe Ihres Instance-Typs. Größere Instance-Typen erfüllen nicht nur die Anforderungen der ausführbaren Dateien Ihrer Gameserver-Software, sondern können auch mehrere and/or Gameserver-Prozesscontainer auf jeder Instance ausführen. Zu den zu berücksichtigenden Faktoren gehören die Kosten (ist es günstiger, ein paar große Instances oder viele kleine Instances zu betreiben). Denken Sie auch darüber nach, wie sich das Hinzufügen oder Entfernen von Instances während der Flottenskalierung oder beim Herunterfahren fehlerhafter Instances auf die Kapazität von Spielsitzungen auswirken könnte. Wenn auf jeder Instanz viele Spieleserverprozesse gleichzeitig ausgeführt werden, kann das Hinzufügen oder Entfernen einer Instanz die Hosting-Kapazität des Spiels erheblich beeinträchtigen.

Weitere Informationen zu Instance-Typen finden Sie unter [Amazon EC2 Instance Types](#).

On-Demand-Instances im Vergleich zu Spot Instances

Amazon EC2 On-Demand-Instances und Spot-Instances bieten dieselbe Hardware und Leistung, unterscheiden sich jedoch in Verfügbarkeit und Kosten.

On-Demand Instances

Sie können eine On-Demand-Instance erwerben, wenn Sie sie benötigen, und sie so lange behalten, wie Sie möchten. On-Demand-Instances haben feste Kosten, d. h. Sie zahlen nur für die Zeit, für die Sie sie nutzen. Es gibt keine langfristigen Verpflichtungen.

Spot Instances

Spot-Instances können durch die Nutzung ungenutzter AWS Rechenkapazität eine kosteneffiziente Alternative zu On-Demand-Instances bieten. Die Preise für Spot-Instances schwanken je nach

Angebot und Nachfrage für jeden Instance-Typ an jedem Standort. AWS kann Spot-Instances mit einer zweiminütigen Benachrichtigung zurückfordern, wenn die Kapazität wieder benötigt wird und Spielsitzungen, die aktiv auf einer zurückgeforderten Instanz laufen, unterbrochen werden.

Amazon GameLift Servers bietet verschiedene Tools, um die Wahrscheinlichkeit von Spot-Unterbrechungen Ihrer Spielsitzungen zu verringern. Ein Spot-Viability-Algorithmus verfolgt historische Daten vom Instance-Typ, um vorherzusehen, wann das Risiko einer Unterbrechung einen kritischen Punkt erreicht, und verhindert, dass neue Spielsitzungen auf Spot-Instances dieses Typs stattfinden. Wenn es zu einer Unterbrechung kommt, können Ihre Spieleserver die Benachrichtigung verwenden, um eine Spielsitzung für Spieler ordnungsgemäß zu beenden.

Beim Hosten von Spielen mit Spot-Flotten muss eine Warteschlange für die Platzierung von Spielsitzungen verwendet werden. In einer Warteschlange können Platzierungen für Spielsitzungen auf der Grundlage der Rentabilität der Spot-Flotte, der Kosten und anderer Faktoren priorisiert werden. In diesen Themen findest du weitere Informationen darüber, wie du Spot für dein Gameserver-Hosting nutzen kannst:

- [Senken Sie die Kosten für das Hosting von Spielen mit Spot-Flotten](#)
- [Eine Warteschlange für Spot-Instances erstellen](#)

Servicekontingente

Sie können die Standard-Servicekontingente Amazon GameLift Servers und den aktuellen Kontingentstatus für Ihre AWS-Konto Nutzung der folgenden Tools einsehen:

- Allgemeine Informationen zu Servicekontingenten für Amazon GameLift Servers finden Sie unter [Amazon GameLift Servers Endpunkte und Kontingente](#) in der Allgemeine AWS-Referenz.
- Eine Liste der verfügbaren Instanztypen pro Standort für Ihr Konto finden Sie auf der Seite mit den [Servicekontingenten](#) der Amazon GameLift Servers Konsole. Auf dieser Seite wird auch die aktuelle Nutzung Ihres Kontos für jeden Instanztyp an jedem Standort angezeigt.
- Führen Sie den Befehl AWS Command Line Interface (AWS CLI) aus, um eine Liste der aktuellen Kontingente Ihres Kontos für Instance-Typen pro Region zu erhalten [describe-ec2-instance-limits](#). Dieser Befehl gibt die Anzahl der aktiven Instances zurück, die Sie in Ihrer Standardregion (oder in einer anderen von Ihnen angegebenen Region) haben.

Während du dich darauf vorbereitest, dein Spiel zu starten, fülle in der [Amazon GameLift Servers Konsole](#) einen Fragebogen zum Start aus. Das Amazon GameLift Servers Team verwendet

den Fragebogen zur Veröffentlichung, um die richtigen Kontingente und Limits für dein Spiel zu ermitteln.

Passen Sie eine Amazon GameLift Servers Containerflotte an

In den Themen in diesem Abschnitt werden einige der optionalen Funktionen für Amazon GameLift Servers verwaltete Container beschrieben. Sie können wählen, ob Sie eine oder alle dieser Funktionen verwenden möchten.

Themen

- [Legen Sie Ressourcenlimits fest](#)
- [Drive Access konfigurieren NVMe](#)
- [Benennen Sie wichtige Container](#)
- [Netzwerkverbindungen konfigurieren](#)
- [Richten Sie Zustandsprüfungen für Container ein](#)
- [Legen Sie Container-Abhängigkeiten fest](#)
- [Konfigurieren Sie eine Containerflotte](#)

Legen Sie Ressourcenlimits fest

Für jede Containergruppe können Sie bestimmen, wie viel Arbeitsspeicher und Rechenleistung die Containergruppe benötigt, um ihre Software auszuführen. Amazon GameLift Servers stützt sich auf diese Informationen, um Ressourcen in der gesamten Containergruppe zu verwalten. Es verwendet diese Informationen auch, um zu berechnen, wie viele Spielserver-Containergruppen ein Flottenimage aufnehmen kann. Sie können auch Grenzwerte für einzelne Container festlegen.

Sie können ein maximales Limit für Speicher und Rechenleistung für eine Containergruppe festlegen. Standardmäßig werden diese Ressourcen von allen Containern in der Gruppe gemeinsam genutzt. Sie können die Ressourcenverwaltung weiter anpassen, indem Sie Grenzwerte für einzelne Container festlegen.

Legen Sie optionale Grenzwerte für einzelne Container fest

Durch die Festlegung containerspezifischer Ressourcenlimits können Sie besser kontrollieren, wie einzelne Container die Ressourcen der Gruppe nutzen können. Wenn Sie keine containerspezifischen Grenzwerte festlegen, teilen sich alle Container in der Gruppe die Gruppenressourcen. Die gemeinsame Nutzung bietet mehr Flexibilität, um Ressourcen dort

einzusetzen, wo sie benötigt werden. Es erhöht auch das Potenzial, dass Prozesse miteinander konkurrieren und zum Ausfall von Containern führen.

Legen Sie für jeden Container eine der folgenden `ContainerDefinition` Eigenschaften fest.

- `MemoryHardLimitMebibytes`— Legt ein maximales Speicherlimit für den Container fest. Wenn der Container dieses Limit überschreitet, führt dies zu einem Neustart.
- `Vcpulimit` — Reservieren Sie eine Mindestmenge an vCPU-Ressourcen für die ausschließliche Verwendung des Containers. Dem Container steht immer die reservierte Menge zur Verfügung. Dieses Minimum kann jederzeit überschritten werden, sofern zusätzliche Ressourcen verfügbar sind. (1024 CPU-Einheiten entsprechen 1 vCPU.)

Legen Sie die Gesamtressourcenlimits für eine Containergruppe fest

Wenn Sie Grenzwerte für einzelne Container festlegen, müssen Sie möglicherweise ändern, wie viel Arbeitsspeicher und vCPU-Ressourcen die Containergruppe benötigt. Ziel ist es, genügend Ressourcen zuzuweisen, um die Leistung des Spieleservers zu optimieren. Amazon GameLift Servers verwendet diese Grenzwerte, um zu berechnen, wie Spielservers-Containergruppen auf einer Flotteninstanz gepackt werden. Sie werden sie auch verwenden, wenn Sie einen Instance-Typ für eine Containerflotte auswählen.

Berechnen den Gesamtspeicher und die vCPU, die für eine Containergruppe benötigt werden. Berücksichtigen Sie dabei Folgendes:

- Was sind all die Prozesse, die auf allen Containern in der Containergruppe ausgeführt werden? Addieren Sie die für diese Prozesse erforderlichen Ressourcen. Beachten Sie alle containerspezifischen Grenzwerte.
- Wie viele Gameserver-Prozesse planen Sie, in jeder Containergruppe gleichzeitig auszuführen? Sie bestimmen dies in Ihrem Gameserver-Container-Image.

Legen Sie auf der Grundlage Ihrer Schätzung der Anforderungen an Containergruppen die folgenden `ContainerGroupDefinition` Eigenschaften fest:

- `TotalMemoryLimitMebibytes`— Legen Sie ein maximales Speicherlimit für die Containergruppe fest. Alle Container in der Gruppe teilen sich den zugewiesenen Speicher. Wenn Sie Grenzwerte für einzelne Container festlegen, muss das Gesamtspeicherlimit gleich oder größer als das höchste containerspezifische Speicherlimit sein.
- `TotalVcpuLimit`— Legen Sie ein maximales vCPU-Limit für die Containergruppe fest. Alle Container in der Gruppe teilen sich die zugewiesenen CPU-Ressourcen. Wenn Sie Grenzwerte für einzelne Container festlegen, muss das gesamte CPU-Limit gleich oder größer als die

Summe aller containerspezifischen CPU-Grenzwerte sein. Es hat sich bewährt, diesen Wert so festzulegen, dass er die Summe der CPU-Grenzwerte für Container verdoppelt.

Beispielszenario

Nehmen wir an, wir definieren eine Spieleserver-Containergruppe mit den folgenden drei Containern:

- Container A ist unser Gameserver-Container. Wir schätzen den Ressourcenbedarf für einen Spieleserver auf 512 MiB und 1024 CPU. Wir planen, dass der Container einen Serverprozess ausführt. Da auf diesem Container unsere wichtigste Software ausgeführt wird, haben wir kein Speicherlimit oder vCPU-Reservelimit festgelegt.
- Container B ist ein Support-Container mit geschätzten Ressourcenanforderungen von 1024 MiB und 1536 CPUs. Wir haben ein Speicherlimit von 2048 MiB und ein CPU-Reservelimit von 1024 CPU festgelegt.
- Container C ist ein weiterer Support-Container. Wir haben ein Festplattenlimit von 512 MiB und ein CPU-Reservelimit von 512 CPU festgelegt.

Anhand dieser Informationen haben wir die folgenden Gesamtgrenzen für die Containergruppe festgelegt:

- Gesamtspeicherlimit: 7680 MiB. Dieser Wert überschreitet das höchste Speicherlimit (1024 MiB).
- Gesamtes CPU-Limit: 13312 CPU. Dieser Wert überschreitet die Summe des CPU-Limits (1024+512 CPU).

Drive Access konfigurieren NVMe

Bei Instanzen vom Typ D wird das NVMe Laufwerk beim Host-Start automatisch in das `/data` Verzeichnis eingebunden. Um Containern den Zugriff auf den SSD-Speicher zu ermöglichen, legen Sie die folgende `ContainerGroupDefinition` Eigenschaft fest: `MountPoints`

- `InstancePath`— Wird auf gesetzt, `/data` um auf das automatisch gemountete NVMe Laufwerk auf der Host-Instance zu verweisen.
- `AccessLevel`— Wählen Sie die passende Zugriffsebene für die Bedürfnisse Ihres Containers (z. B. `READ_ONLY` oder `READ_WRITE`).
- `ContainerPath`— (Optional) Geben Sie den Pfad an, unter dem der Instanzpfad innerhalb des Containers bereitgestellt werden soll. Wenn nicht angegeben, wird standardmäßig der Instanzpfad verwendet.

Weitere Informationen zu Bereitstellungspunkten finden Sie [ContainerMountPoint](#) in der Amazon GameLift Servers API Reference.

Benennen Sie wichtige Container

Für eine Containergruppe pro Instanz kennzeichnen Sie jeden Container als essenziell oder nicht essenziell. Containergruppen pro Instanz müssen über mindestens einen Container für die Unterstützung von essenzieller Bedeutung verfügen. Der Essential-Container erledigt die wichtige Arbeit der Container-Gruppe. Es wird erwartet, dass der Essential-Container immer läuft. Wenn dies fehlschlägt, wird die gesamte Containergruppe neu gestartet.

Setzen Sie `Essential` die `ContainerDefinition` Eigenschaft für jeden Container entweder auf `true` oder `false`.

Netzwerkverbindungen konfigurieren

Sie können den Netzwerkzugriff so anpassen, dass externer Datenverkehr eine Verbindung zu jedem Container in einer Containerflotte herstellen kann. Sie müssen beispielsweise Netzwerkverbindungen zu dem Container herstellen, auf dem Ihre Spieleserverprozesse ausgeführt werden, damit Spieleclients Ihrem Spiel beitreten und es spielen können. Spieleclients stellen über Ports und IP-Adressen eine Verbindung zu Spieleservern her.

In einer Containerflotte besteht keine direkte Verbindung zwischen einem Client und einem Server. Intern hört ein Prozess in einem Container auf einem Containerport ab. Extern stellt der eingehende Verkehr über einen Verbindungsport eine Verbindung zu einer Flotteninstanz her. Amazon GameLift Servers verwaltet die Zuordnungen zwischen internen Container-Ports und nach außen gerichteten Verbindungsports, sodass eingehender Datenverkehr an den richtigen Prozess auf der Instance weitergeleitet wird.

Amazon GameLift Servers bietet eine zusätzliche Kontrollebene für Ihre Netzwerkverbindungen. Jede Containerflotte verfügt über eine Einstellung für Eingangsberechtigungen, mit der Sie den Zugriff auf jeden nach außen gerichteten Verbindungsport kontrollieren können. Sie könnten beispielsweise die Berechtigungen für alle Verbindungsports entfernen, um den gesamten Zugriff auf die Container der Flotte zu unterbinden.

Sie können die Eingangsberechtigungen, Verbindungs- und Container-Ports einer Flotte aktualisieren.

 Warning

Wenn Sie einen benutzerdefinierten InstanceConnectionPortRange Wert angeben InstanceInboundPermissions, Amazon GameLift Servers wird keiner der beiden Werte für Ihre Flotte mehr verwaltet. Sie müssen beide Felder festlegen, um undefiniertes Verhalten zu vermeiden.

Legen Sie die Portbereiche für Container fest

Konfigurieren Sie Container-Portbereiche als Teil jeder Containerdefinition. Dies ist ein erforderlicher Parameter für eine Containergruppendefinition. Sie müssen genügend Ports konfigurieren, um alle gleichzeitig laufenden Prozesse zu unterstützen, die externen Zugriff benötigen. Einige Container benötigen keine Ports.

Dein Gameserver-Container, auf dem deine Gameserver laufen, benötigt einen Port für jeden gleichzeitig laufenden Gameserver-Prozess. Der Gameserver-Prozess überwacht den zugewiesenen Port und meldet ihn an. Amazon GameLift Servers

Stellen Sie die Verbindungs-Portbereiche ein

Konfigurieren Sie Ihre Containerflotte mit einer Reihe von Verbindungsanschlüssen. Verbindungsports bieten externen Zugriff auf die Flotteninstanzen, auf denen Ihre Container laufen. Amazon GameLift Servers weist Verbindungsports zu und ordnet sie nach Bedarf Container-Ports zu.

Amazon GameLift Servers berechnet standardmäßig die Anzahl der Ports, die für alle Containergruppen erforderlich sind, und legt einen Portbereich fest, um diese zu berücksichtigen. Es wird dringend empfohlen, Amazon GameLift Servers berechnete Werte zu verwenden, die aktualisiert werden, wenn Sie Aktualisierungen für eine Containergruppendefinition bereitstellen. Wenn Sie die Verbindungs-Portbereiche anpassen müssen, folgen Sie der folgenden Anleitung.

Wenn Sie eine Containerflotte erstellen, definieren Sie einen Verbindungsbereich (siehe [ContainerFleet: InstanceConnectionPortRange](#)). Stellen Sie sicher, dass der Bereich über genügend Anschlüsse verfügt, um jedem Containerhafen zugeordnet zu werden, der für alle Container in beiden Containergruppen der Flotte definiert ist. Verwenden Sie die folgende Formel, um die mindestens erforderlichen Verbindungsanschlüsse zu berechnen:

```
[Total number of container ports defined for containers in the game  
server container group] * [Number of game server container groups per
```


`instance] + [Total number of container ports defined for containers in the per-instance container group]`

Es hat sich bewährt, die Mindestanzahl an Verbindungsanschlüssen zu verdoppeln.

 Note

Die Anzahl der Verbindungsports kann möglicherweise die Anzahl der Spielserver-Containergruppen pro Instanz einschränken. Wenn eine Flotte nur über genügend Verbindungsports für eine Spielserver-Containergruppe pro Instanz verfügt, Amazon GameLift Servers wird nur eine Spielserver-Containergruppe bereitgestellt, auch wenn die Instances über genügend Rechenleistung für mehrere Spielserver-Containergruppen verfügen.

Legen Sie Berechtigungen für eingehende Nachrichten fest

Eingehende Berechtigungen steuern den externen Zugriff auf eine Containerflotte, indem sie angeben, welche Verbindungsports für eingehenden Verkehr geöffnet werden sollen. Sie können diese Einstellung verwenden, um den Netzwerkzugriff einer Flotte nach Bedarf ein- und auszuschalten.

Amazon GameLift Servers berechnet standardmäßig die Anzahl der Ports, die für alle Containergruppen erforderlich sind, und legt einen Portbereich fest, um diese zu berücksichtigen. Es wird dringend empfohlen, Amazon GameLift Servers berechnete Werte zu verwenden, die aktualisiert werden, wenn Sie Aktualisierungen für eine Containergruppendefinition bereitstellen. Wenn Sie die Verbindungs-Portbereiche anpassen müssen, folgen Sie der folgenden Anleitung.

Wenn Sie eine Containerflotte erstellen, definieren Sie eine Reihe von Berechtigungen für eingehende Sendungen (siehe [ContainerFleet: InstanceInboundPermissions](#)). Die Ports für eingehende Genehmigungen sollten den Verbindungs-Portbereichen der Flotte entsprechen.

 Note

Da die Container-Ports nach dem Zufallsprinzip ausgewählt werden InstanceConnectionPortRange, InstanceConnectionPortRange sollten alle Ports in durch Ports in abgedeckt werden, um sicherzustellen, dass Sitzungsverbindungen hergestellt werden können InstanceInboundPermissions

Beispielszenario

Dieses Beispiel zeigt, wie alle drei Netzwerkverbindungseigenschaften festgelegt werden.

- Die Gameserver-Containergruppe unserer Flotte besteht aus einem Container, auf dem ein Spielservers-Prozess ausgeführt wird.

In der Definition der Spielservers-Containergruppe legen wir den `PortConfiguration` Parameter für diesen Container wie folgt fest:

```
"PortConfiguration": {  
  "ContainerPortRanges": [ { "FromPort": 10, "ToPort": 20, "Protocol": "TCP" } ]  
}
```

- Unsere Flotte hat auch eine Containergruppe pro Instanz mit einem Container. Es hat einen Prozess, der Netzwerkzugriff benötigt. In der Instanz-Container-Definition legen wir den `PortConfiguration` Parameter für diesen Container wie folgt fest:

```
"PortConfiguration": {  
  "ContainerPortRanges": [ { "FromPort": 25, "ToPort": 25, "Protocol": "TCP" } ] } }
```

- Unsere Flotte ist mit 20 Spielservers-Containergruppen pro Flotteninstanz konfiguriert. Anhand dieser Informationen können wir die Formel verwenden, um die Anzahl der benötigten Verbindungsports zu berechnen:
 - Minimum: 21 Ports [1 Spielservers-Container-Ports * 20 Spielservers-Containergruppen pro Instanz + 1 Container-Port pro Instanz]
 - Bewährtes Verfahren: 42 Anschlüsse [Mindestanzahl Anschlüsse x 2]

Bei der Erstellung der Containerflotte haben wir den `InstanceConnectionPortRange` Parameter wie folgt festgelegt:

```
"InstanceConnectionPortRange": { "FromPort": 1010, "ToPort": 1071 }
```

- Wir möchten den Zugriff auf alle verfügbaren Verbindungsports ermöglichen. Bei der Erstellung der Containerflotte haben wir den `InstanceInboundPermissions` Parameter wie folgt festgelegt:

```
"InstanceInboundPermissions": [  
  { "FromPort": 1010, "ToPort": 1071, "IpRange": "10.24.34.0/23", "Protocol":  
    "TCP" } ]
```

Richten Sie Zustandsprüfungen für Container ein

Ein Container wird automatisch neu gestartet, wenn ein Terminalfehler auftritt, und er wird nicht mehr ausgeführt. Wenn ein Container als unverzichtbar erachtet wird, fordert er die gesamte Containergruppe auf, neu zu starten.

Alle Gameserver-Container werden automatisch als unverzichtbar angesehen.

Unterstützungscontainer können als unverzichtbar eingestuft werden, sie müssen jedoch über einen Mechanismus zur Meldung von Gesundheitsproblemen verfügen. Sie können auch Integritätsprüfungen für Support-Container einrichten, die nicht unbedingt erforderlich sind.

Sie können zusätzliche benutzerdefinierte Kriterien definieren, um den Zustand von Containern zu messen, und diese Kriterien mit einer Integritätsprüfung testen. Um eine Container-Integritätsprüfung einzurichten, können Sie sie in einem Docker-Container-Image oder in Ihrer Container-Definition definieren. Wenn Sie in der Container-Definition eine Integritätsprüfung festlegen, überschreibt diese alle Einstellungen im Container-Image.

Legen Sie die folgenden `SupportContainerDefinition` Eigenschaften für eine Container-Integritätsprüfung fest:

- **Command**— Stellen Sie einen Befehl bereit, der einen Aspekt des Zustands des Containers überprüft. Sie entscheiden, nach welchen Kriterien der Gesundheitszustand gemessen werden soll. Der Befehl muss zu einem Ausgangswert von 1 (fehlerhaft) oder 0 (fehlerhaft) führen.
- **StartPeriod**— Geben Sie eine anfängliche Verzögerung an, bis Fehler bei der Integritätsprüfung gezählt werden. Diese Verzögerung gibt dem Container Zeit, seine Prozesse zu booten.
- **Interval**— Entscheiden Sie, wie oft der Health Check-Befehl ausgeführt werden soll. Wie schnell möchten Sie einen Container-Fehler erkennen und beheben?
- **Timeout**— Entscheiden Sie, wie lange Sie auf Erfolg oder Misserfolg warten möchten, bevor Sie den Befehl zur Integritätsprüfung erneut ausführen. Wie lange sollte es dauern, bis der Befehl zur Integritätsprüfung abgeschlossen ist?
- **Retries**— Wie oft sollte der Health-Check-Befehl wiederholt werden, bevor ein Fehler registriert wird?

Legen Sie Container-Abhängigkeiten fest

Innerhalb jeder Containergruppe können Sie Abhängigkeiten zwischen Containern basierend auf dem Containerstatus festlegen. Eine Abhängigkeit wirkt sich darauf aus, wann der abhängige Container je nach Status eines anderen Containers gestartet oder heruntergefahren werden kann.

Ein wichtiger Anwendungsfall für Abhängigkeiten ist die Erstellung von Start- und Shutdown-Sequenzen für die Container-Gruppe.

Sie möchten beispielsweise, dass Container A zuerst gestartet und erfolgreich abgeschlossen wird, bevor Container B und C gestartet werden. Um dies zu erreichen, erstellen Sie zunächst eine Abhängigkeit für Container B von Container A mit der Bedingung, dass Container A erfolgreich abgeschlossen werden muss. Erstellen Sie dann eine Abhängigkeit für Container C von Container A mit derselben Bedingung. Die Startsequenzen werden beim Herunterfahren in umgekehrter Reihenfolge ausgeführt.

Konfigurieren Sie eine Containerflotte

Wenn Sie eine Containerflotte erstellen, sollten Sie die folgenden Entscheidungspunkte berücksichtigen. Die meisten dieser Punkte hängen von Ihrer Container-Architektur und -Konfiguration ab.

Entscheiden Sie, wo Sie Ihre Flotte einsetzen möchten

Im Allgemeinen möchten Sie Ihre Flotten geografisch in der Nähe Ihrer Spieler einsetzen, um die Latenz zu minimieren. Du kannst deine Containerflotte auf jede Art einsetzen, die AWS-Region das Amazon GameLift Servers unterstützt. Wenn du denselben Spielservers an weiteren geografischen Standorten einsetzen möchtest, kannst du der Flotte entfernte Standorte hinzufügen, einschließlich AWS-Regionen Local Zones. Bei einer Flotte mit mehreren Standorten kannst du die Kapazität an jedem Flottenstandort unabhängig anpassen. Weitere Informationen zu unterstützten Flottenstandorten finden Sie unter [Amazon GameLift Servers Servicestandorte](#).

Erwägen Sie [UDP-Ping-Beacons](#) die Erfassung von Netzwerklatenzdaten an verschiedenen geografischen Standorten, um die Latenz zwischen Spielergeräten und potenziellen Flottenstandorten zu antizipieren. Diese speziellen Endpunkte akzeptieren UDP-Nachrichten anstelle von herkömmlichen ICMP-Pings und bieten präzise Latenzmessungen, die Sie bei der Auswahl der optimalen Flottenstandorte unterstützen.

Wählen Sie einen Instance-Typ und eine Größe für Ihre Flotte

Amazon GameLift Servers unterstützt eine Vielzahl von Amazon EC2 EC2-Instance-Typen, die alle für die Verwendung mit einer Containerflotte verfügbar sind. Die Verfügbarkeit und der Preis des Instance-Typs variieren je nach Standort. Eine nach Standort gefilterte Liste der unterstützten Instanztypen finden Sie in der Amazon GameLift Servers Konsole (unter Ressourcen, Instanz und Dienstkontingente).

Bei der Auswahl eines Instanztyps sollten Sie zunächst die Instanzfamilie berücksichtigen. Instance-Familien bieten verschiedene Kombinationen von CPU-, Arbeitsspeicher-, Speicher- und Netzwerkfunktionen. Erfahren Sie mehr über [EC2-Instance-Familien](#). Innerhalb jeder Familie haben Sie eine Reihe von Instance-Größen zur Auswahl. Beachten Sie bei der Auswahl einer Instance-Größe die folgenden Aspekte:

- Was ist die Mindestinstanzgröße, die Ihren Workload unterstützen kann? Verwenden Sie diese Informationen, um zu kleine Instance-Typen zu eliminieren.
- Welche Instance-Typgrößen eignen sich gut für Ihre Container-Architektur? Idealerweise sollten Sie eine Größe wählen, die mehrere Kopien Ihrer Spielservers-Containergruppe aufnehmen kann und dabei nur wenig Speicherplatz verschwendet.
- Welche Skalierungsgranularität ist für dein Spiel sinnvoll? Beim Skalieren der Flottenkapazität müssen Instanzen hinzugefügt oder entfernt werden, und jede Instanz steht für die Fähigkeit, eine bestimmte Anzahl von Spielsitzungen zu hosten. Überlegen Sie, wie viel Kapazität Sie mit jeder Instanz hinzufügen oder entfernen möchten. Wenn die Nachfrage der Spieler von Minute zu Minute um Tausende schwankt, kann es sinnvoll sein, sehr große Instanzen zu verwenden, die Hunderte oder Tausende von Spielsitzungen hosten können. Im Gegensatz dazu bevorzugen Sie möglicherweise eine detailliertere Skalierungssteuerung bei kleineren Instance-Typen.
- Gibt es je nach Größe Kosteneinsparungen? Möglicherweise stellen Sie fest, dass die Kosten bestimmter Instance-Typen je nach Standort je nach Verfügbarkeit variieren.

Legen Sie weitere optionale Flotteneinstellungen fest

Bei der Konfiguration einer Containerflotte können Sie die folgenden optionalen Funktionen verwenden:

- Richte deine Spieleserver so ein, dass sie auf andere AWS Ressourcen zugreifen können. Siehe [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).
- Schütze Spielsitzungen mit aktiven Spielern davor, während eines Scale-Down-Events vorzeitig beendet zu werden.
- Beschränken Sie die Anzahl der Spielsitzungen, die eine Person innerhalb eines begrenzten Zeitraums auf der Flotte erstellen kann.

Senken Sie die Kosten für das Hosting von Spielen mit Spot-Flotten

Beim Hosten von Multiplayer-Spieleservern mithilfe von Amazon GameLift Servers Managed Hosting können Spot-Instances eine kostengünstige Alternative zu On-Demand-Instances darstellen. Das Spot-Preismodell bietet dieselbe Hardware und Leistung wie On-Demand-Preismodell, bietet jedoch potenziell erhebliche Kosteneinsparungen (bis zu 70-90%). Sie haben jedoch eine Einschränkung: Wenn Kapazitäten wieder AWS benötigt werden, kann es diese Instanzen mit einer zweiminütigen Unterbrechungsbenachrichtigung zurückfordern.

Amazon GameLift Servers mindert das Risiko von Unterbrechungen beim Hosting von Spieleservern. Amazon GameLift Servers prognostiziert die Wahrscheinlichkeit von Unterbrechungen bei Spot-Instance-Typen und verhindert, dass Spielsitzungen auf gefährdeten Instances stattfinden. Wenn es in seltenen Fällen zu einer Unterbrechung kommt, können Sie mit der Benachrichtigung eine Spielsitzung für Spieler ordnungsgemäß beenden.

Wie Amazon GameLift Servers funktioniert mit Spot-Flotten

Wenn Sie Spot-Flotten für das Hosten von Spielen einrichten, werden Ihre Instance-Typen und Standorte Ihrer Spot-Flotte Amazon GameLift Servers fortlaufend auf die Rentabilität des Spiele-Hostings überprüft.

- Der Spot-Viability-Algorithmus analysiert aktuelle Verfügbarkeitsmuster und historische Unterbrechungsraten von Spot-Instance-Typen nach Standort.
- Auf der Grundlage dieser Analyse Amazon GameLift Servers werden Spot-Instance-Typen und Standorte identifiziert, an denen ein inakzeptables Potenzial für Unterbrechungen von Spielsitzungen besteht. Es werden die folgenden Maßnahmen ergriffen:
 - Es markiert die Kombination aus Instance-Typ und Standort als vorübergehend nicht praktikabel.
 - Dadurch werden alle Standorte der Spot-Flotte, die nicht funktionieren, bei der Platzierung neuer Spielsitzungen nicht berücksichtigt. Daher werden Spielsitzungen nur an Standorten der Spot-Flotte abgehalten, bei denen die Wahrscheinlichkeit eines unterbrechungsfreien Spieleserver-Hostings hoch ist.
 - Dadurch wird der Standort der Spot-Flotte vorhandener Instances entzogen, auch wenn sie AWS nicht zurückgewonnen werden. Sie zahlen also nicht für Instances, die nicht für das Hosten von Spielen verwendet werden können. Wenn der Schutz von Spielsitzungen aktiviert ist, werden Instances erst nach Abschluss der aktiven Spielsitzungen heruntergefahren.
- Amazon GameLift Servers überprüft die Instance-Typen und Standorte deiner Spot-Flotte kontinuierlich auf ihre Rentabilität beim Hosten von Spielen. Wenn ein zuvor nicht funktionsfähiger

Instance-Typ auf der Grundlage aktualisierter historischer Daten wieder funktionsfähig wird, können Sie die Spot-Flotte erneut skalieren und wieder Amazon GameLift Servers Spielsitzungen damit durchführen.

Designüberlegungen

Wenn Sie Ihre Lösung für die Verwendung von Spot-Flotten entwerfen, sollten Sie die folgenden Aspekte berücksichtigen:

- Beurteilen Sie die Länge Ihrer Spielsitzungen — Die durchschnittliche Länge Ihrer Spielsitzungen kann sich darauf auswirken, wie gut Spot für Ihr Spiel funktioniert. Bei kürzeren Spielsitzungen sorgt die schnellere Bearbeitungszeit dafür, dass Spielsitzungen auf brauchbaren Instanztypen laufen, die auf den neuesten historischen Daten basieren. Längere Spielsitzungen werden weiterhin auf Instanztypen ausgeführt, ohne dass aktuelle Daten zur Rentabilität ausgewertet werden, wodurch das Risiko einer Unterbrechung im Laufe der Zeit steigt.
- Evaluieren Sie die Verfügbarkeit von Instance-Typen — Nicht jeder Flottenstandort bietet jeden Instance-Typ als Spot. Verwenden Sie bei der Auswahl eines Instance-Typs für Ihre Spot-Flotte das Tool zur Erstellung von Amazon GameLift Servers Konsolenflotten, um Spot-Instance-Typen an den benötigten Standorten zu finden. Mit diesem Tool können Sie Ihre Flottenstandorte auswählen und dann die Verfügbarkeit der Instance-Typen an diesen Standorten einsehen.
- Spot-Flotten mit mehreren Standorten erstellen — Sie können Spot-Flotten mit mehreren Standorten erstellen. Eine einzelne Spot-Flotte mit mehreren Standorten stellt Instances mit demselben Instance-Typ in mehreren AWS-Regionen oder Local Zones bereit. Der Spot-Viability-Algorithmus bewertet die Rentabilität sowohl auf der Grundlage des Instance-Typs als auch auf der Grundlage des Standorts. Wenn ein Standort einer Spot-Flotte als nicht rentabel eingestuft wird, hat dies keine Auswirkungen auf andere Standorte in der Flotte, die dennoch für Spielsitzungen genutzt werden können.
- Erstellen Sie Warteschlangen mit unterschiedlicher Spot-Flotte — Wenn Sie Spot-Flotten für das Hosten von Spielen verwenden, müssen Sie eine Warteschlange für die Platzierung von Spielsitzungen einrichten. Bei jeder neuen Anfrage für eine Spielsitzung sucht die Warteschlange nach verfügbaren Ressourcen zum Hosten von Spielen und wählt die bestmögliche Option aus. Mit Spot-Flotten benötigen Sie eine Warteschlange, die mehrere Flotten durchsuchen kann, die sich sowohl in Bezug auf Standorte als auch Instance-Typen unterscheiden, und Sie möchten mindestens eine On-Demand-Flotte als Backup-Kapazität einbeziehen. Eine gut konzipierte Warteschlange mit mehreren Flotten, die verschiedene Platzierungsoptionen bietet, ist äußerst widerstandsfähig gegen Unterbrechungen, Verlangsamungen und Ausfälle. Weitere Hinweise zur

Gestaltung einer Warteschlange für Spot finden Sie unter. [Eine Warteschlange für Spot-Instances erstellen](#)

- Gehen Sie elegant mit Unterbrechungen um — Richten Sie Ihren Spielservers so ein, dass die Spieler im Falle einer Spot-Unterbrechung so wenig wie möglich beeinträchtigt werden. Wenn eine Spot-Instance AWS zurückerobert wird, Amazon GameLift Servers leitet die Kündigungsbenachrichtigung mithilfe der Server-SDK-Callback-Funktion an alle betroffenen Serverprozesse weiter. `onProcessTerminate()` Ihr Spiel muss diesen Callback implementieren, um die Spielsitzung ordnungsgemäß beenden zu können. Weitere Informationen finden Sie unter [Reagieren Sie auf eine Benachrichtigung zum Herunterfahren des Serverprozesses](#).

 Note

AWS bemüht sich nach besten Kräften, die Benachrichtigung bereitzustellen, bevor es eine Instance zurückfordert, aber es ist möglich, dass die Spot-Instance zurückgefordert wird, bevor die AWS Warnung eintrifft. Du solltest deinen Spielservers auch darauf vorbereiten, unerwartete Unterbrechungen zu bewältigen.

- Konfigurieren Sie die auto-scaling für Ihre Backup-Flotten, um den Service während einer Spot-Unterbrechung aufrechtzuerhalten. Die auto-scaling mit Zielverfolgung sorgt für einen Kapazitätspuffer und skaliert automatisch bei Bedarf. Mit der auto-scaling beginnen Backup-Flotten (Spot oder On-Demand), ihre Kapazität zu erhöhen, sobald sie mehr Anfragen für Spielsitzungen erhalten.

Um verlorene Kapazitäten schnell zu ersetzen, wenn eine Spot-Flotte nicht mehr rentabel ist, kann ein benutzerdefinierter Skalierungsmechanismus anhand verfügbarer Warteschlangen- und Flottenmetriken eine schnelle Skalierung der Backup-Flotten einleiten. Ermitteln Sie anhand von Kennzahlen wie, und, wann Spot-Flotten nicht mehr rentabel werden.

`FirstChoiceOutOfCapacity` `FirstChoiceNotViable` `PercentAvailableGameSessions`
Schätzen Sie den Bedarf an Ersatzkapazitäten ab, indem Sie aktuelle `PlacementsStarted` Metrikdaten analysieren. Nach der Skalierung der Backup-Flotten zur Deckung des unmittelbaren Bedarfs kann die normale auto-scaling die Kontrolle übernehmen.

- Integration mit FlexMatch — Wenn Ihre Lösung einen FlexMatch Matchmaker verwendet, gibt es keine besonderen Anforderungen für Spot-Flotten. Sie können einen Matchmaker so konfigurieren, dass er eine Warteschlange mit Spot-Flotten verwendet. Amazon GameLift Servers priorisiert automatisch die Platzierung von Spielen in allen Spot- und On-Demand-Flotten, auch bei der Platzierung neuer Spielsitzungen und beim Auffüllen leerer Spielerplätze in bestehenden Spielsitzungen.

Optimiere die Laufzeitkonfiguration des Spieleservers auf verwalteten Amazon GameLift Servers

Sie können die Laufzeitkonfiguration einer verwalteten EC2-Flotte so einrichten, dass mehrere Spielserverprozesse pro Instanz ausgeführt werden. Dadurch werden Ihre Hosting-Ressourcen effizienter genutzt.

So verwaltet eine Flotte mehrere Prozesse

Amazon GameLift Servers verwendet die Laufzeitkonfiguration einer Flotte, um die Art und Anzahl der Prozesse zu bestimmen, die auf jeder Instanz ausgeführt werden sollen. Eine Laufzeitkonfiguration enthält mindestens eine Serverprozesskonfiguration, die eine ausführbare Datei für einen Spieleserver darstellt. Sie können zusätzliche Serverprozesskonfigurationen definieren, um andere Arten von Prozessen im Zusammenhang mit Ihrem Spiel auszuführen. Jede Serverprozesskonfiguration enthält die folgenden Informationen:

- Den Dateinamen und den Pfad einer ausführbaren Datei in Ihrem Spiele-Build
- (Optional) Parameter, die beim Start an den Prozess weitergeleitet werden
- Die Anzahl der Prozesse, die gleichzeitig ausgeführt werden sollen.

Wenn eine Instanz in der Flotte aktiviert wird, startet sie die in der Laufzeitkonfiguration definierten Serverprozesse. Bei mehreren Prozessen wird der Start Amazon GameLift Servers der einzelnen Prozesse gestaffelt. Serverprozesse haben eine begrenzte Lebensdauer. Amazon GameLift Servers startet am Ende neue Prozesse, um die Anzahl und den Typ der Serverprozesse beizubehalten, die in der Laufzeitkonfiguration definiert sind.

Sie können die Laufzeitkonfiguration jederzeit ändern, indem Sie Serverprozesskonfigurationen hinzufügen, ändern oder entfernen. Jede Instanz sucht regelmäßig nach Aktualisierungen der Laufzeitkonfiguration der Flotte, um die Änderungen zu implementieren. So werden Änderungen an der Amazon GameLift Servers Laufzeitkonfiguration übernommen:

1. Die Instanz sendet eine Anfrage Amazon GameLift Servers für die neueste Version der Laufzeitkonfiguration an.
2. Die Instanz vergleicht ihre aktiven Prozesse mit der neuesten Laufzeitkonfiguration und führt dann Folgendes aus:
 - Wenn die aktualisierte Laufzeitkonfiguration einen Serverprozesstyp entfernt, werden aktive Serverprozesse dieses Typs weiter ausgeführt, bis sie enden. Die Instanz ersetzt diese Serverprozesse nicht.

- Wenn die aktualisierte Laufzeitkonfiguration die Anzahl der gleichzeitigen Prozesse für einen Serverprozesstyp verringert, werden überschüssige Serverprozesse dieses Typs weiter ausgeführt, bis sie enden. Die Instanz ersetzt diese überschüssigen Serverprozesse nicht.
- Wenn die aktualisierte Laufzeitkonfiguration einen neuen Serverprozesstyp hinzufügt oder die Anzahl der gleichzeitigen Prozesse für einen vorhandenen Typ erhöht, startet die Instanz neue Serverprozesse bis zum Amazon GameLift Servers Maximum. In diesem Fall startet die Instanz neue Serverprozesse, sobald bestehende Prozesse enden.

Optimieren Sie eine Flotte für mehrere Prozesse

Gehen Sie wie folgt vor, um mehrere Prozesse in einer Flotte zu verwenden:

- [Erstellen Sie einen Build](#), der die ausführbaren Dateien des Spielservers enthält, die Sie für eine Flotte bereitstellen möchten, und laden Sie den Build dann hoch. Amazon GameLift Servers
Alle Spielservers in einem Build müssen auf derselben Plattform laufen und das Server-SDK für verwenden. Amazon GameLift Servers
- Erstellen Sie eine Laufzeitkonfiguration mit einer oder mehreren Serverprozesskonfigurationen und mehreren gleichzeitigen Prozesse.
- Integrieren Sie Spieleclients mit der AWS SDK-Version 2016-08-04 oder höher.

Um die Leistung Ihres Fuhrparks zu optimieren, empfehlen wir Ihnen, wie folgt vorzugehen:

- Bewältigen Sie Szenarien, in denen Serverprozesse heruntergefahren Amazon GameLift Servers werden, sodass Prozesse effizient wiederverwendet werden können. Beispiel:
 - Füge deinem Spielservercode ein Verfahren zum Herunterfahren hinzu, das die Server-API `ProcessEnding()` aufruft.
 - Implementieren Sie die Callback-Funktion `OnProcessTerminate()` in Ihrem Spielservercode, um Kündigungsanfragen von Amazon GameLift Servers zu bearbeiten.
- Stellen Sie sicher, dass Amazon GameLift Servers fehlerhafte Serverprozesse heruntergefahren und neu gestartet werden. Melde den Integritätsstatus an, Amazon GameLift Servers indem du die `OnHealthCheck()` Rückruffunktion in deinen Spielservercode implementierst. Amazon GameLift Servers fährt Serverprozesse, die in drei aufeinanderfolgenden Berichten als fehlerhaft gemeldet wurden, automatisch herunter. Wenn Sie dies nicht implementieren `OnHealthCheck()`, wird Amazon GameLift Servers davon ausgegangen, dass ein Serverprozess fehlerfrei ist, es sei denn, der Prozess reagiert nicht auf eine Kommunikation.

Wählen Sie die Anzahl der Prozesse pro Instanz

Beachten Sie bei der Entscheidung, wie viele Prozesse gleichzeitig auf einer Instance ausgeführt werden sollen, Folgendes:

- Amazon GameLift Servers begrenzt jede Instance auf eine [maximale Anzahl an gleichzeitigen Prozessen](#). Die Summe aller gleichzeitigen Prozesse für die Serverprozesskonfigurationen einer Flotte darf dieses Kontingent nicht überschreiten.
- Um ein akzeptables Leistungsniveau aufrechtzuerhalten, kann der Amazon EC2 EC2-Instance-Typ die Anzahl der Prozesse einschränken, die gleichzeitig ausgeführt werden können. Testen Sie verschiedene Konfigurationen für Ihr Spiel, um die richtige Anzahl von Prozessen für Ihren bevorzugten Instance-Typ zu finden.
- Amazon GameLift Servers führt nicht mehr Prozesse gleichzeitig aus als die konfigurierte Gesamtzahl. Das bedeutet, dass der Übergang von der vorherigen Laufzeitkonfiguration zur neuen Konfiguration schrittweise erfolgen kann.

Arbeite mit dem Amazon GameLift Servers Agenten

Der Amazon GameLift Servers Agent überwacht den Betrieb der Spielserver-Prozesse auf Ihren Amazon GameLift Servers Flotten. Der Agent wird auf jedem Computer in einer Flotte eingesetzt und bietet automatisiertes Prozessmanagement, Hosting-Management und Protokollierung der Rechenleistung. Um den Agenten verwenden zu können, muss dein Spielserver-Build in das Server-SDK für Amazon GameLift Servers Version 5.x oder höher integriert sein.

Der Amazon GameLift Servers Agent ist extern für Flotten verfügbar, bei denen es sich nicht um verwaltete Amazon GameLift Servers EC2 Flotten handelt. (Verwaltete EC2 Flotten erledigen die Aufgaben des Agenten automatisch.) Sie können sich dafür entscheiden, Amazon GameLift Servers Flotten, einschließlich Anywhere-Flotten, mit oder ohne den Agenten zu betreiben. Ohne den Agenten müssen Sie eine alternative Lösung für die Ausführung der erforderlichen Aufgaben bereitstellen.

Bei der Bereitstellung auf einem Computer sollte der Amazon GameLift Servers Agent gestartet werden, bevor irgendwelche Spielserverprozesse gestartet werden. Beim Start führt der Agent die folgenden Aufgaben aus:

- Registriert die Rechenleistung mithilfe der [RegisterCompute](#) API bei einer Amazon GameLift Servers Anywhere-Flotte.

- Ruft die [GetComputeAuthToken](#) API auf, um ein Autorisierungstoken abzurufen, und speichert es zur Verwendung durch Serverprozesse, die auf dem Computer ausgeführt werden.
- Legt die WebSocket URL-Umgebungsvariable für die Berechnung fest und stellt eine WebSocket Verbindung zum Amazon GameLift Servers Dienst her.
- Fordert die neueste Version der Laufzeitkonfiguration der Flotte vom Amazon GameLift Servers Service an.
- Startet und stoppt Serverprozesse gemäß den Anweisungen zur Laufzeitkonfiguration.

Quellcode und Build-Anweisungen für den Amazon GameLift Servers Agenten sind im [Amazon GameLift ServersAgent](#) verfügbar GitHub.

Über den Agenten

Der Amazon GameLift Servers Agent ist so konzipiert, dass er die folgenden Aufgaben für Ihre Flotten übernimmt:

Prozessmanagement

- Startet neue Serverprozesse, wie in den Laufzeitanweisungen definiert. Der Agent verwendet möglicherweise eine benutzerdefinierte Laufzeitkonfiguration, die zusammen mit dem Agenten bereitgestellt wird. Alternativ können Sie eine RuntimeConfiguration als Teil Ihrer Flottendefinition angeben. Dieser Ansatz hat den Vorteil, dass Sie die Laufzeitkonfiguration der Flotte jederzeit ändern können. Der Agent fordert regelmäßig aktualisierte Laufzeitkonfigurationen vom Amazon GameLift Servers Service an.
- Überwacht die Aktivierungen von Serverprozessen und beendet Prozesse, wenn sie nicht rechtzeitig aktiviert werden.
- Sendet Heartbeats an. Amazon GameLift Servers Wenn der Agent keine Heartbeats sendet, wird die Rechenleistung möglicherweise als veraltet markiert.
- Meldet, Amazon GameLift Servers wann ein Serverprozess endet. Amazon GameLift Servers verwendet diese Informationen, um die Verfügbarkeit des Spielservers für die Platzierung von Spielsitzungen zu überwachen.
- Sendet Flottenereignisse für Serverprozesse aus, darunter:
 - SERVER_PROCESS_INVALID_PATH: Die Startparameter für den Spieleserverprozess waren falsch konfiguriert.

- **SERVER_PROCESS_TERMINATED_UNHEALTHY:** Der Spielserver-Prozess hat innerhalb von 3 Minuten nach der Aktivierung keinen gültigen Gesundheitscheck gemeldet und wurde daher beendet.
- **SERVER_PROCESS_FORCE_TERMINATED:** Der Spielserver-Prozess wurde nicht ordnungsgemäß beendet, nachdem `OnProcessTerminate()` er innerhalb von 30 Sekunden gesendet wurde.
- **SERVER_PROCESS_CRASHED:** Ein Gameserver-Prozess ist aus irgendeinem Grund abgestürzt.

Computerverwaltung

- Empfängt Nachrichten vom Amazon GameLift Servers Dienst, um die Rechenleistung herunterzufahren.
- Fordert auf, die Berechnung zu beenden durch Amazon GameLift Servers.

Protokollierung

- Lädt Protokolle in einen Amazon S3 S3-Bucket in Ihrem AWS Konto hoch.

Zusammenfassung einer Amazon GameLift Servers Flottenbezeichnung mit einem Alias

Ein Amazon GameLift Servers Alias wird verwendet, um ein Hosting-Ziel zu abstrahieren. Hosting-Ziele geben Amazon GameLift Servers an, wo nach verfügbaren Ressourcen gesucht werden muss, um eine neue Spielsitzung für Spieler zu veranstalten. Aliase sind in den folgenden Szenarien nützlich:

- Wenn dein Spiel keine Warteschlangen mit mehreren Flotten für die Platzierung von Spielsitzungen verwendet, fordert es neue Spielsitzungen an, indem du eine Amazon GameLift Servers Flotten-ID angibst. Während der Laufzeit eines Spiels tauschst du die Flotte mehrmals aus, um einen Serveraufbau zu aktualisieren, die Hosting-Hardware und das Betriebssystem zu aktualisieren oder Leistungsprobleme zu lösen. Verwende einen Alias, um die Flotten-ID zu abstrahieren, sodass du den Spielerverkehr nahtlos von einer bestehenden Flotte auf die neue übertragen kannst.
- Wenn du etwas anderes tun möchtest, als eine neue Spielsitzung zu erstellen, wenn ein Spielclient eine anfordert. Vielleicht möchtest du Spieler, die einen out-of-date Client verwenden, zu einer Upgrade-Website weiterleiten.

Ein Alias muss eine Routing-Strategie angeben. Es gibt zwei Typen. Eine einfache Routing-Strategie leitet Spielerverkehr an eine bestimmte Flotten-ID weiter, die Sie aktualisieren können, um den Verkehr umzuleiten. Eine Terminal-Routing-Strategie leitet eine Nachricht zurück an den Client, anstatt eine neue Spielsitzung zu erstellen. Du kannst die Routing-Strategie eines Alias jederzeit ändern.

Wenn du eine Warteschlange für die Platzierung von Spielsitzungen verwendest, benötigst du keinen Alias, um den Traffic umzuleiten, wenn du eine Flotte ersetzt. Mit einer Warteschlange kannst du einfach die neue Flotte hinzufügen und die alte Flotte entfernen. Diese Aktion ist für Spieler nicht sichtbar, da neue Anfragen für Spielsitzungen automatisch mit der neuen Flotte erfüllt werden. Sie hat keine Auswirkungen auf bestehende Spielsitzungen. Sie können die Ziele von Warteschlangen entweder anhand einer Flotten-ID oder eines Alias identifizieren.

Themen

- [Erstelle eine Amazon GameLift Servers alias](#)

Erstelle eine Amazon GameLift Servers alias

In diesem Thema wird beschrieben, wie Sie ein erstellen Amazon GameLift Servers Alias zur Verwendung bei der Platzierung von Spielsitzungen.

Um einen Alias zu erstellen

Verwenden Sie entweder Amazon GameLift Servers Konsole oder AWS Command Line Interface (AWS CLI), um einen Alias zu erstellen.

Console

In der [Amazon GameLift Servers Konsole](#), verwenden Sie den Navigationsbereich, um die Seite Aliase zu öffnen.

1. Wählen Sie „Alias erstellen“.
2. Geben Sie einen Aliasnamen ein. Wir empfehlen, aussagekräftige Merkmale in den Aliasnamen aufzunehmen, um die Anzeige einer Aliasliste zu erleichtern.
3. Geben Sie nach Bedarf eine Alias-Beschreibung ein.
4. Wählen Sie eine Routing-Strategie für den Alias.

- a. Wenn Sie sich für eine einfache Routing-Strategie entscheiden, wählen Sie eine Flotten-ID aus der Liste aus, die Sie diesem Alias zuordnen möchten. Die Liste enthält alle Flotten, die derzeit ausgewählt AWS-Region sind. Sie müssen einen Alias in derselben Region wie die Flotte erstellen.
 - b. Wenn Sie sich für eine Terminal-Routing-Strategie entscheiden, geben Sie den gewünschten Zeichenkettenwert ein Amazon GameLift Servers um als Antwort auf eine Anfrage für eine Spielsitzung zu einem Spielclient zurückzukehren. Eine Anfrage mit einem Terminalalias löst eine Ausnahme aus, bei der die Nachricht eingebettet ist.
5. (Optional) Fügen Sie der Alias-Ressource Tags hinzu. Jeder Tag (Markierung) besteht aus einem Schlüssel und einem optionalen Wert, beides können Sie bestimmen. Weisen Sie AWS Ressourcen, die Sie sinnvoll kategorisieren möchten, Tags zu, z. B. nach Zweck, Eigentümer oder Umgebung. Wählen Sie für jedes Tag, das Sie hinzufügen möchten, die Option Neues Tag hinzufügen aus.
6. Wenn Sie bereit sind, die neue Flotte einzusetzen, wählen Sie Create.

AWS CLI

Verwenden Sie den [create-alias](#)Befehl, um einen Alias zu erstellen. Amazon GameLift Servers erstellt die Alias-Ressource in Ihrem aktuellen Standard AWS-Region (oder Sie können ein --region-Tag hinzufügen, um einen anderen anzugeben AWS-Region).

Geben Sie mindestens einen Aliasnamen und eine Routing-Strategie an. Für eine einfache Routing-Strategie geben Sie die ID einer Flotte in derselben Region wie der Alias an. Geben Sie für eine Terminal-Routing-Strategie eine Nachrichtenzeichenfolge an.

Verwaltung einer Amazon GameLift Servers Hosting-Lösung (Markteinführung und danach)

Dieser Abschnitt enthält Anleitungen zur langfristigen Verwaltung Ihrer Amazon GameLift Servers Ressourcen. Der erfolgreiche Start Ihres Multiplayer-Spiels ist erst der Anfang. Die Verwaltung einer Produktions-Hosting-Lösung erfordert kontinuierliche Aufmerksamkeit in Bezug auf Leistung, Kostenoptimierung, Sicherheitsupdates und Spielererlebnis. Dieser Abschnitt enthält Anleitungen zu den betrieblichen Aspekten des Betriebs Amazon GameLift Servers in der Produktionsumgebung, von den Aktivitäten am ersten Tag der Markteinführung bis hin zur langfristigen Wartung und Optimierung.

Ganz gleich, ob Sie sich auf die Veröffentlichung vorbereiten, ein Live-Spiel verwalten oder eine etablierte Hosting-Lösung optimieren, diese Themen helfen Ihnen dabei, ein zuverlässiges, kostengünstiges Game-Hosting aufrechtzuerhalten, das mit Ihrer Spielerbasis skaliert.

Verwalte Game-Hosting-Ressourcen mit Amazon GameLift Servers

Verwalte Amazon GameLift Servers Ressourcen wie Builds, Flotten, Warteschlangen für Spielsitzungen und Matchmaker. Sie können die Amazon GameLift Servers Konsole verwenden, um Ressourcenlisten zu scannen oder Ressourcendetails einschließlich Metriken anzuzeigen. Sie können Ressourceninformationen auch mit dem AWS SDK oder der AWS CLI für abrufen Amazon GameLift Servers

Amazon GameLift ServersFunktionen der Konsole

Verwende die Amazon GameLift Servers Konsole, um deine Spiele-Hosting-Ressourcen und deine laufenden Hosting-Aktivitäten nahezu in Echtzeit einzusehen und zu verwalten. Die Konsole bietet eine grafische Oberfläche für die meisten Funktionen der Service-API fürAmazon GameLift Servers. Sie können die Konsole verwenden, um:

- Verwenden Sie das Dashboard für einen Snapshot auf hoher Ebene. Sie können die Anzahl und den aktuellen Status all Ihrer Amazon GameLift Servers Hosting-Ressourcen einsehen und Links folgen, um Details zu einzelnen Ressourcen zu erhalten.
- Verwalte einzelne Hosting-Ressourcen. Sie können alle Amazon GameLift Servers Ressourcen erstellen, anzeigen und löschen und ihre veränderbaren Eigenschaften aktualisieren. Sie können auch bestimmte Arten von Hosting-Aktivitäten wie Ereignisse und Leistungskennzahlen anzeigen.

- Interagiere mit den Aktivitäten im Spiel und in der Spielsitzung. Du kannst die Spielsitzung und die Aktivität der Spielsitzungen nach Flotte verfolgen und diese Informationen zur Behebung von Problemen mit Spielsitzungen verwenden. Sehen Sie sich Details zu einer Spielsitzung an, sehen Sie sich Spielsitzungen für jede Spielsitzung an und suchen Sie nach Spieleraktivitäten über mehrere Spielsitzungen hinweg. Sie können auch einzelne Spielsitzungen nach Bedarf beenden.

Themen

- [Hosting-Dashboard in der Amazon GameLift Servers Konsole](#)
- [Spieleserver-Builds](#)
- [Flotten in der Konsole Amazon GameLift Servers](#)
- [Aliase in der Konsole Amazon GameLift Servers](#)
- [Warteschlangen für Spielsitzungen in der Konsole Amazon GameLift Servers](#)

Hosting-Dashboard in der Amazon GameLift Servers Konsole

Verwenden Sie das Amazon GameLift Servers Konsolen-Dashboard, um einen allgemeinen Überblick über den aktuellen Status der Amazon GameLift Servers Hosting-Ressourcen in Ihrem AWS Konto zu erhalten. Das Amazon GameLift ServersDashboard bietet einen Überblick über Folgendes:

- Die Anzahl der Builds mit den Status Bereit, Initialisiert und Fehlgeschlagen. Wählen Sie Builds anzuzeigen, um weitere Informationen zu Builds in Ihrer aktuellen Region zu erhalten.
- Die Anzahl der Flotten in allen Status. Wählen Sie „Flotten anzeigen“, um weitere Informationen zu Flotten in Ihrer aktuellen Region zu erhalten.
- Ihre aktuellen Ressourcen.
- Ankündigungen neuer Funktionen und Dienste.

So öffnen Sie das Amazon GameLift Servers-Dashboard

- Wählen Sie in der [Amazon GameLift ServersKonsole](#) im Navigationsbereich Dashboard aus.

Über das Dashboard können Sie:

- Bereite dein Spiel für den Start vor, indem du „Für den Start vorbereiten“ auswählst und den entsprechenden Fragebogen zur Veröffentlichung ausfüllst.

- Fordere als Vorbereitung oder als Reaktion auf Produkteinführungen eine Erhöhung der Servicekontingenten an, indem du „Servicekontingente anzeigen“ auswählst.
- Um Blogbeiträge und detaillierte Informationen zu neuen Funktionen anzuzeigen, klicken Sie auf den Link im Feature-Spotlight.

GameLift > Dashboard

Dashboard

Build status overview

Viewing data for all builds in N. Virginia region.

Ready
1

Initialized
0

Failed
0

[View builds](#)

Fleet status overview

Viewing data for all fleets in N. Virginia region.

Active
0

Deleting
0

In progress
0

New
0

Error
0

Terminated
0

[View fleets](#)

Resources (1)

[View service quotas](#)

Resource type	Count
Builds	1
Scripts	0
Fleets	0
Aliases	0
Queues	0
Matchmaking rule sets	0
Matchmaking configurations	0

Prepare for your game launch

[Learn more](#)

Fill out a launch questionnaire

Fill out our game launch questionnaire and email it to the GameLift launch team to ensure a smooth launch. The GameLift launch team will verify your GameLift setup and service limits, preparing you for launch.

[Prepare to launch](#)

Features spotlight

Updates on features available in N. Virginia region

March 22, 2022

Updates to Amazon GameLift FlexMatch for greater flexibility

October 28, 2021

New Asia Pacific (Osaka) region and Graviton2 support for Amazon GameLift

Spieleserver-Builds

Die Build-Ressource steht für Ihre Gameserver-Software. Sie laden Ihr Build-Paket Amazon GameLift Servers zur Bereitstellung in verwaltete EC2 Flotten hoch.

Informationen zu Spielserver-Builds findest du in der Amazon GameLift Servers Konsole oder mithilfe des oder AWS SDK für Amazon GameLift Servers.

Console

Auf der Build-Seite der [Amazon GameLift Servers Konsole](#) kannst du Informationen über alle Spielserver-Builds einsehen und verwalten, die du Amazon GameLift Servers für den Einsatz auf verwalteten EC2 Flotten hochgeladen hast. Wählen Sie im Navigationsbereich „Hosting“, „Verwaltet EC2“ und „Builds“ aus.

Auf der Seite Builds werden die folgenden Informationen für jeden Build angezeigt. Sie können den Tabelleninhalt mit dem Einstellungen-Tool nach Bedarf anpassen (siehe das



Symbol in der oberen rechten Ecke der Tabelle). Benutzerdefinierte Einstellungen werden für den Benutzer Ihres AWS Benutzerkontos gespeichert und automatisch angewendet, wenn Sie diese Seite aufrufen.

Note

Auf der Seite Builds werden nur Builds in Ihrer aktuellen AWS Region angezeigt.

- Name — Der Name, der dem hochgeladenen Build zugeordnet ist.
- Status — Der Status des Builds. Zeigt eine von drei Statusmeldungen an:
 - Initialisiert — Der Upload wurde nicht gestartet oder ist noch im Gange.
 - Bereit — Der Build ist bereit für die Flottenerstellung.
 - Fehlgeschlagen — Beim Build wurde das Timeout überschritten, bevor die Binärdateien Amazon GameLift Servers empfangen wurden.
- Erstellungszeit — Datum und Uhrzeit, zu denen Sie den Build hochgeladen haben. Amazon GameLift Servers
- Build-ID — Die eindeutige ID, die dem Build beim Upload zugewiesen wurde.
- Version — Die Versionsbezeichnung, die dem hochgeladenen Build zugeordnet ist.
- Betriebssystem — Das Betriebssystem, auf dem der Build ausgeführt wird. Das Build-Betriebssystem bestimmt, welches Betriebssystem auf den Instanzen einer Flotte Amazon GameLift Servers installiert wird.

- **Größe** — Die Größe der Build-Datei, in die hochgeladen wurde, in Megabyte (MB). Amazon GameLift Servers
- **Flotten** — Die Anzahl der Flotten, die mit dem Build eingesetzt wurden.

Auf dieser Seite können Sie die folgenden Schritte ausführen:

- Zeigen Sie die Build-Details an. Wählen Sie den Namen eines Builds, um die zugehörige Build-Detailseite zu öffnen.
- Erstellen Sie eine neue Flotte von einem Build. Wählen Sie einen Build und dann Flotte erstellen aus.
- Filtern und sortieren Sie die Build-Liste. Verwenden Sie die Steuerelemente oben in der Tabelle.
- Löschen Sie einen Build. Wählen Sie einen Build aus und klicken Sie dann auf Löschen.

Details erstellen

Wählen Sie auf der Seite Builds den Namen eines Builds aus, um die zugehörige Detailseite zu öffnen. Im Abschnitt „Übersicht“ der Detailseite werden dieselben Informationen zur Build-Zusammenfassung angezeigt wie auf der Seite „Builds“. Im Abschnitt „Flotten“ wird eine Liste der Flotten angezeigt, auf denen der Build ausgeführt wird, einschließlich derselben Übersichtsinformationen wie auf der Seite „[Flotten](#)“.

AWS SDK

Verwenden Sie die folgenden AWS CLI-Befehle, um Informationen zu dieser Ressource abzurufen:

- [ListBuild](#)
- [DescribeBuild](#)

Aktualisiere einen Spielserver-Build

Wenn Sie Ihren Gameserver-Build für Amazon GameLift Servers verwaltetes EC2-Hosting bereitstellen, laden Sie Ihre Gameserver-Software hoch und erstellen eine Amazon GameLift Servers Build-Ressource. Nachdem Sie einen Amazon GameLift Servers Build erstellt haben, können Sie die Metadaten des Builds aktualisieren, aber Sie können die Build-Dateien selbst nicht aktualisieren. Um Updates für deinen Gameserver-Build bereitzustellen, musst du einen neuen Satz von Dateien

hochladen und mit dem Befehl AWS CLI [upload-build](#)command einen neuen Amazon GameLift Servers Build erstellen. Alternativ können Sie den [create-build](#)Befehl verwenden, um einen neuen Build aus einem Amazon S3 S3-Bucket hochzuladen, den Sie kontrollieren. Stellen Sie dann den neuen Build bereit, indem Sie eine neue Flotte dafür erstellen.

Sie können die Metadaten eines Builds aktualisieren, einschließlich des Namens und der Beschreibung. Verwenden Sie für diese Aufgaben die Amazon GameLift Servers Konsole oder den [update-build](#) AWS CLI Befehl.

Automatisieren Sie Ihre Spiele-Build-Updates

Befolge diese Tipps, um den Prozess der Aktualisierung von Spielserver-Builds für Amazon GameLift Servers verwaltete Flotten zu automatisieren und zu optimieren:

- Nutze Warteschlangen für Spielsitzungen und tausche Flotten nach Bedarf aus. Wenn du Anfragen für Spielsitzungen an sendestAmazon GameLift Servers, gib statt einer bestimmten Flotte eine Warteschlange für Spielsitzungen an. Mit Warteschlangen kannst du Flotten mit einem neuen Build hinzufügen und alte Flotten nach Bedarf entfernen. Weitere Informationen finden Sie unter [Platzierung der Spielsitzungen konfigurieren](#).
- Verwende Aliase, um Spieler auf einen neuen Spiel-Build zu übertragen. Wenn du Anfragen für Spielsitzungen an sendestAmazon GameLift Servers, gib statt einer Flotten-ID einen Flotten-Alias an. Weitere Informationen finden Sie unter [Erstelle eine Amazon GameLift Servers alias](#).
- Für eine iterative Entwicklung eingerichtet. Erkunden Sie während der Spieleentwicklung die Optionen für die Einrichtung einer gehosteten Testumgebung, die eine schnelle iterative Entwicklung unterstützt. Siehe [Für iterative Entwicklung mit Amazon GameLift Servers Anywhere einrichten](#).

Probiere diese Ressourcen aus dem [Amazon GameLift ServersToolkit](#) auf Github aus:

Schnelles Build-Update-Tool (nur für die Entwicklung)

Mit diesem Tool können Sie Spielserver-Builds ändern, die bereits auf Computern in einer verwalteten EC2-Flotte bereitgestellt werden, wodurch Sie Zeit bei schnellen Entwicklungsiterationen sparen. Das Tool bietet mehrere Optionen: Sie können den gesamten Spielbuild ersetzen oder bestimmte Dateien ändern, und Sie können festlegen, wie die Spielserverprozesse nach den Updates neu gestartet werden. Du kannst es auch verwenden, um alle Computer in einer Flotte zu aktualisieren oder einzelne Computer als Ziel zu verwenden.

Besuchen Sie das Amazon GameLift Servers Toolkit-Repo auf Github, um das [schnelle Build-Update-Tool](#) auf Github herunterzuladen und mehr darüber zu erfahren, wie Sie es verwenden können.

Beispielskript für die Bereitstellung in der Produktion

Dieses Skript veranschaulicht, wie Sie den Prozess der Aktualisierung von Spielserver-Builds automatisieren können, die auf verwalteten EC2-Flotten in der Produktion bereitgestellt werden. Um dieses Skript verwenden zu können, muss Ihre Amazon GameLift Servers Hosting-Lösung Aliase verwenden, um die Flotte zu abstrahieren. IDs Das Beispielskript behandelt die folgenden grundlegenden Schritte: Laden Sie einen aktualisierten Build hoch, erstellen Sie einen neuen Build und setzen Sie ihn in einer neuen Flotte ein, leiten Sie den Spielerverkehr von einer vorhandenen Flotte zur neuen Flotte um und löschen Sie die alte Flotte. Passen Sie das Beispielskript an Ihre spezifischen Bereitstellungsanforderungen an.

Besuchen Sie das Amazon GameLift Servers Toolkit-Repo auf Github, um das [Beispielskript für die Produktionsbereitstellung](#) auf Github herunterzuladen und mehr darüber zu erfahren, wie Sie es verwenden können.

Flotten in der Konsole Amazon GameLift Servers

Die Flottenressource stellt eine Reihe von Rechenmaschinen dar, die auf Ihren Spieleservern bereitgestellt werden und diese ausführen. Abhängig von Ihrer Hosting-Lösung verfügen Sie möglicherweise über verwaltete EC2 Flotten, verwaltete Containerflotten, Anywhere-Flotten oder eine Kombination aus beiden.

Informationen zu Flotten können Sie in der Amazon GameLift Servers Konsole oder mithilfe des oder SDK für anzeigen. AWS Amazon GameLift Servers

Console

Amazon GameLift ServersUnter deinem Konto findest du Informationen zu allen Flotten, die für das Hosten deiner Spiele erstellt wurden. AWS Suchen Sie in der linken Navigationsleiste der Konsole nach der gewünschten Hosting-Option (Anywhere, Managed EC2, Managed Containers) und wählen Sie Fleets aus.

Für jede Hosting-Option werden auf der Seite „Flotten“ Flotten aufgeführt, die sich in der aktuell ausgewählten Region befinden. AWS Auf der Seite „Flotten“ können Sie eine neue Flotte erstellen oder zusätzliche Details zu einer Flotte anzeigen. Die [Detailseite](#) einer Flotte enthält

Nutzungsinformationen, Messwerte, Spielsitzungsdaten und Spielersitzungsdaten. Du kannst auch einen Flottendatensatz bearbeiten oder eine Flotte löschen.

Auf einer Flottenseite werden standardmäßig die folgenden zusammenfassenden Informationen angezeigt. Sie können den Tabelleninhalt mit dem Einstellungen-Tool nach Bedarf anpassen (siehe das



Symbol in der oberen rechten Ecke der Tabelle). Benutzerdefinierte Einstellungen werden für den Benutzer Ihres AWS Benutzerkontos gespeichert und automatisch angewendet, wenn Sie diese Seite aufrufen.

- ID — Eine Kennung, die der Flotte zugewiesen wurde. Diese ID ist innerhalb des Ortes, AWS-Region an dem die Flotte erstellt wurde, eindeutig.
- Name — Ein freundlicher Name, der der Flotte gegeben wurde.
- Status — Der Status der Flotte: „Neu“, „Wird heruntergeladen“, „Im Aufbau“ und „Aktiv“.
- Erstellungszeit — Datum und Uhrzeit der Erstellung der Flotte.

 Note

Für Flotten, die vor mehr als 90 Tagen erstellt wurden, wird ein Warnsymbol angezeigt. Als bewährte Methode empfehlen wir, die Flotten alle 30 Tage auszutauschen, um eine sichere up-to-date Laufzeitumgebung für Ihre gehosteten Spieleserver aufrechtzuerhalten. Anleitungen finden Sie unter [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#).

- Flottenart — Die Verfügbarkeit der Instanzen, die zum Hosten deiner Spiele verwendet werden, was sich möglicherweise auf die Hosting-Kosten auswirken kann. Eine verwaltete Flotte kann On-Demand-Instances (immer verfügbar) oder Spot-Instances (Verfügbarkeit variiert) verwenden.
- Aktive Instances — Die Anzahl der EC2 Instances, die für die Flotte verwendet werden.
- Gewünschte Instances — Die Anzahl der EC2 Instances, die aktiv bleiben sollen.
- Spielsitzungen — Die Anzahl der aktiven Spielsitzungen, die in der Flotte ausgeführt werden. Die Daten werden um fünf Minuten verzögert.
- Spielersitzungen — Die Anzahl der aktiven Spielersitzungen in der Flotte. Die Daten werden um fünf Minuten verzögert.

AWS SDK

Verwenden Sie die folgenden AWS CLI-Befehle, um Informationen zu dieser Ressource abzurufen:

- Verwaltete EC2 Flotten
 - [ListFleets](#)
 - [ListCompute](#)
 - [DescribeFleetAttributes](#)
 - [DescribeCompute](#)
 - [DescribeFleetCapacity](#)
 - [DescribeFleetUtilization](#)
- Verwaltete Containerflotten
 - [ListContainerFleets](#)
 - [DescribeContainerFleet](#)
 - [ListContainerGroupDefinitions](#)
 - [DescribeContainerGroupDefinition](#)
- - [ListFleets](#)
 - [ListCompute](#)
 - [DescribeFleetAttributes](#)
 - [DescribeCompute](#)

Eine Amazon GameLift Servers Flottenkonfiguration aktualisieren

Verwenden Sie die Amazon GameLift Servers Konsole oder die AWS CLI, um Ihre Flotteneinstellungen zu aktualisieren, entfernte Standorte zu ändern oder eine Flotte zu löschen. Bei verwalteten Flotten kannst du den Gameserver-Build oder den Instanztyp einer Flotte nicht ändern. Stattdessen musst du die Flotte ersetzen.

Fast Build Update Tool (nur für Entwicklungszwecke)

Bei verwalteten EC2 Flotten musst du, um ein Build-Update für den Spieleserver bereitzustellen, jeden neuen Build hochladen Amazon GameLift Servers und eine neue Flotte

Mit dem Fast Build Update Tool kannst du diese Schritte während der Entwicklung umgehen, was dir Zeit spart und eine schnellere Iteration der Entwicklung ermöglicht. Mit diesem Tool kannst du deine Spiel-Build-Dateien schnell auf allen Computern einer vorhandenen Flotte aktualisieren. Das Tool bietet mehrere Optionen: Sie können einen kompletten Spiel-Build ersetzen oder 6 bestimmte Dateien ändern, und Sie können festlegen, wie die Spielserver-Prozesse nach den Updates neu gestartet werden sollen. Du kannst es auch verwenden, um einzelne Computer in einer Flotte zu aktualisieren.

Um das Fast Build Update Tool herunterzuladen und mehr über seine Verwendung zu erfahren, besuchen Sie das Amazon GameLift Servers Toolkit-Repo für [das Fast Build Update Tool](#) auf Github.

Sie können veränderbare Flottenattribute, Porteinstellungen und Laufzeitkonfigurationen über die Amazon GameLift Servers Konsole oder die AWS CLI aktualisieren. Informationen zum Ändern der Skalierungsgrenzen finden Sie unter [Automatische Skalierung der Flottenkapazität mit Amazon GameLift Servers](#).

Console

1. Wählen Sie in der [Amazon GameLift ServersKonsole](#) im Navigationsbereich Fleets aus.
2. Wählen Sie die Flotte aus, die Sie aktualisieren möchten. Eine Flotte muss sich im ACTIVE Status befinden, bevor Sie sie bearbeiten können.
3. Wählen Sie auf der Flottendetailseite in einem der folgenden Abschnitte die Option Bearbeiten aus.
 - Flotteneinstellungen
 - Ändern Sie Flottenattribute wie Name (Name) und Description (Beschreibung).
 - Fügen Sie Metrikgruppen hinzu oder entfernen Sie sie, die Amazon CloudWatch verwendet, um aggregierte Amazon GameLift Servers Kennzahlen für mehrere Flotten zu verfolgen.
 - Aktualisieren Sie die Einstellungen für die Begrenzung der Ressourcenerstellung.
 - Aktivieren oder deaktivieren Sie den Spielsitzungsschutz.
 - Laufzeitkonfiguration — Sie können jede der folgenden Einstellungen Ihrer Laufzeitkonfigurationen ändern und Laufzeitkonfigurationen hinzufügen oder entfernen.
 - Ändere den Startpfad deines Spielervers.

- Optionale Startparameter hinzufügen, entfernen oder ändern.
 - Ändere die Anzahl der gleichzeitigen Prozesse, die deine Spieleserver ausführen.
 - Aktivierung von Spielsitzungen — Ändere, wie Serverprozesse ausgeführt und Spielsitzungen veranstaltet werden sollen, indem du die Anzahl gleichzeitiger Spielsitzungen und das Timeout für neue Aktivierungen aktualisierst.
 - EC2 Porteinstellungen — Aktualisieren Sie die IP-Adressen und Portbereiche, die eingehenden Zugriff auf die Flotte ermöglichen.
4. Wählen Sie Bestätigen, um die Änderungen zu speichern.

AWS CLI

Verwenden Sie die folgenden AWS CLI-Befehle, um eine Flotte zu aktualisieren:

- [update-fleet-attributes](#)
- [update-fleet-port-settings](#)
- [update-runtime-configuration](#)

Flottenstandorte aktualisieren

Sie können die Remote-Standorte einer Flotte über die Amazon GameLift Servers Konsole oder die AWS CLI hinzufügen oder entfernen. Sie können die Heimatregion einer Flotte nicht ändern.

Console

1. Wählen Sie in der [Amazon GameLift ServersKonsole](#) im Navigationsbereich die Option Flotten aus.
2. Wählen Sie die Flotte aus, die Sie aktualisieren möchten. Eine Flotte muss sich im ACTIVE Status befinden, bevor Sie sie bearbeiten können.
3. Wählen Sie auf der Flottendetailseite den Tab Standorte aus, um die Standorte der Flotte anzuzeigen.
4. Um neue Remote-Standorte hinzuzufügen, wählen Sie Hinzufügen und wählen Sie die Standorte aus, an denen Sie Instances bereitstellen möchten. Diese Liste enthält keine Instances, bei denen der Instance-Typ der Flotte nicht verfügbar ist.
5. Wenn neue Standorte ausgewählt sind, wählen Sie Hinzufügen aus. Amazon GameLift Servers fügt die neuen Standorte der Liste hinzu, wobei der Status auf gesetzt ist NEW.

Amazon GameLift Servers beginnt dann mit der Bereitstellung einer Instanz an jedem hinzugefügten Standort und bereitet sie für das Hosten von Spielsitzungen vor.

6. Um bestehende Remote-Standorte aus der Flotte zu entfernen, verwenden Sie die Kontrollkästchen, um einen oder mehrere aufgelistete Standorte auszuwählen.
7. Wählen Sie Entfernen, wenn eine oder mehrere Flotten ausgewählt sind. Die entfernten Standorte verbleiben in der Liste, wobei der Status auf DELETING gesetzt ist. Amazon GameLift Servers beginnt dann mit dem Beenden der Aktivität am entfernten Standort. Wenn es aktive Instanzen gibt, die Spielsitzungen hosten, Amazon GameLift Servers verwendet das Verfahren zur Kündigung des Spieleservers, um Spielsitzungen ordnungsgemäß zu beenden, Spieleserver zu beenden und Instanzen herunterzufahren.

AWS CLI

Verwenden Sie die folgenden AWS CLI-Befehle, um die Flottenstandorte zu aktualisieren:

- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Löschen Sie eine Flotte

Sie können eine Flotte löschen, wenn Sie sie nicht mehr benötigen. Durch das Löschen einer Flotte werden alle Daten, die mit Spiel- und Spielersitzungen verknüpft sind, sowie die gesammelten Metrikdaten dauerhaft entfernt. Als Alternative können Sie die Flotte beibehalten, Auto Scaling deaktivieren und die Flotte manuell auf 0 Instances skalieren.

Note

Wenn die Flotte über eine VPC-Peering-Verbindung verfügt, fordern Sie zunächst die Autorisierung an, indem Sie anrufen. [CreateVpcPeeringAuthorization](#) Amazon GameLift Servers löscht die VPC-Peering-Verbindung beim Löschen der Flotte.

Sie können entweder die Amazon GameLift Servers Konsole oder das AWS CLI-Tool verwenden, um eine Flotte zu löschen.

Console

1. Wählen Sie in der [Amazon GameLift ServersKonsole](#) im Navigationsbereich Fleets aus.
2. Wählen Sie die Flotte aus, die Sie löschen möchten. Sie können nur Flotten mit dem ERROR Status ACTIVE oder löschen.
3. Wählen Sie Löschen aus.
4. Bestätigen Sie im Dialogfeld Flotte löschen den Löschvorgang, indem Sie Folgendes eingeben**delete**.
5. Wählen Sie Löschen aus.

AWS CLI

Verwenden Sie den folgenden AWS CLI-Befehl, um eine Flotte zu löschen:

- [delete-fleet](#)

Flottendetails in der Amazon GameLift Servers Konsole

Rufen Sie über das Dashboard oder die Flottenseite eine Flottendetailseite auf, indem Sie den Flottennamen auswählen.

Auf der Seite mit den Flottendetails können Sie die folgenden Aktionen ausführen:

- Aktualisieren Sie die Attribute, Porteeinstellungen und die Laufzeitkonfiguration einer Flotte.
- Flottenstandorte hinzufügen oder entfernen.
- Ändern Sie die Flottenkapazitätseinstellungen.
- Stellen Sie die auto Skalierung der Zielverfolgung ein oder ändern Sie sie.
- Löschen einer Flotte.

Details

Flotteneinstellungen

- Flotten-ID — Eine Kennung, die der Flotte zugewiesen wurde. Diese ID ist innerhalb des Ortes, AWS-Region an dem die Flotte erstellt wurde, eindeutig.
- Flottenname — Ein freundlicher Name, der der Flotte gegeben wurde.

- **ARN** — Eine eindeutige Kennung, die dieser Flotte zugewiesen wurde. Der ARN einer Flotte identifiziert sie als Amazon GameLift Servers Ressource und spezifiziert die Region und das AWS Konto.
- **Beschreibung** — Eine kurze identifizierbare Beschreibung der Flotte.
- **Status** — Aktueller Status der Flotte, der „Neu“, „Wird heruntergeladen“, „Im Aufbau“ und „Aktiv“ lauten kann.
- **Erstellungszeit** — Datum und Uhrzeit der Erstellung der Flotte.

 Note

Für Flotten, die vor mehr als 90 Tagen erstellt wurden, wird ein Warnsymbol angezeigt. Als bewährte Methode empfehlen wir, die Flotten alle 30 Tage auszutauschen, um eine sichere up-to-date Laufzeitumgebung für Ihre gehosteten Spieleserver aufrechtzuerhalten. Anleitungen finden Sie unter [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#).

- **Rechenart** — Die Art der Rechenleistung, die zum Hosten deiner Spiele verwendet wird. Bei einer Flotte kann es sich um eine verwaltete EC2, verwaltete Container - oder Anywhere-Flotte handeln.
- **Betriebssystem** — Das Amazon Machine Image (AMI), das für jede Instance in der Flotte bereitgestellt wird. Die AMI-Version ist die neueste Version, die zum Zeitpunkt der Erstellung der Flotte verfügbar war.
- **Zertifikatstyp** — Der Status der Fähigkeit der Flotte, ein TLS-Zertifikat zur Authentifizierung eines Spieleservers und zur Verschlüsselung der gesamten client/server Kommunikation zu verwenden.
- **Flottentyp** — Die Verfügbarkeit der Instanzen, die zum Hosten deiner Spiele verwendet werden, was sich möglicherweise auf die Hosting-Kosten auswirken kann. Eine Flotte kann On-Demand-Instances (immer verfügbar) oder Spot-Instances (Verfügbarkeit variiert) verwenden.
- **EC2 Instance-Typ** — EC2 [Amazon-Instance-Typ](#), der bei der Erstellung für die Flotte ausgewählt wurde.
- **Instanzrolle ARN** — Eine AWS IAM-Rolle, die den Zugriff auf Ihre anderen AWS Ressourcen verwaltet, sofern eine solche bei der Flottenerstellung bereitgestellt wurde.
- **Anbieter von Anmeldeinformationen für Instanzrollen** — Eine AWS IAM-Rolle, die den Zugriff auf Ihre anderen AWS Ressourcen verwaltet, sofern diese bei der Flottenerstellung bereitgestellt wurden.
- **Name der Metrikgruppe** — Die Gruppe, die verwendet wird, um Metriken für mehrere Flotten zu aggregieren.

- Schutzrichtlinie zur Skalierung von Spielen — Gibt an, ob der Spielsitzungsschutz der Flotte aktiviert ist, wodurch verhindert wird, dass aktive Spielsitzungen während eines Scale-Down-Ereignisses vorzeitig beendet werden.
- Maximale Anzahl von Spielsitzungen pro Spieler — Die maximale Anzahl von Sitzungen, die ein Spieler während des Richtlinienzeitraums erstellen kann.
- Versicherungszeitraum — Der Zeitraum, der verwendet wird, um die Anzahl der erstellten Sitzungen eines Spielers zu begrenzen.

Einzelheiten erstellen

Im Abschnitt Build-Details wird der Build angezeigt, der auf der Flotte gehostet wird. Wählen Sie den Build-Namen aus, um die vollständige Build-Detailseite zu sehen.

Laufzeitkonfiguration

Im Abschnitt Runtime-Konfiguration werden die Serverprozesse angezeigt, die auf jeder Instance gestartet werden sollen. Dazu gehören der Pfad für die ausführbare Datei des Spiel-Servers sowie optionale Startparameter.

Aktivierung der Spielsitzung

Im Abschnitt Aktivierung der Spielsitzung wird die Anzahl der Serverprozesse angezeigt, die gleichzeitig gestartet werden, und wie lange es dauert, bis der Prozess aktiviert wird, bevor er beendet wird.

EC2 Port-Einstellungen

Im Abschnitt Ports werden die Verbindungsberechtigungen der Flotte angezeigt, einschließlich der IP-Adressen und der Porteneinstellungsbereiche.

Metriken

Die Registerkarte Metrics (Metriken) zeigt eine grafische Darstellung der Flottenmetriken im Zeitverlauf. Weitere Informationen zur Verwendung von Metriken in Amazon GameLift Servers finden Sie unter [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#).

Ereignisse

Die Registerkarte Events bietet ein Protokoll aller Ereignisse für die Flotte, einschließlich des Ereigniscodes, der Nachricht und des Zeitstempels. Weitere Informationen finden Sie in der Amazon GameLift Servers API-Referenz zu den [Ereignissen](#).

Skalierung

Die Registerkarte Skalierung enthält Informationen zur Flottenkapazität, einschließlich des aktuellen Status und der Kapazitätsänderungen im Laufe der Zeit. Es bietet auch Tools zur Aktualisierung von Kapazitätsgrenzen und zur Verwaltung der auto Skalierung.

Skalierung der Kapazität

Sehen Sie sich die aktuellen Flottenkapazitätseinstellungen für jeden Flottenstandort an. Weitere Informationen zur Änderung von Grenzwerten und Kapazitäten finden Sie unter [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#).

- **AWS Standort** — Name eines Standorts, an dem Flotteninstanzen bereitgestellt werden.
- **Status** – Hosting-Status des Flottenstandorts. Der Standortstatus muss ACTIVE sein, um Spiele hosten zu können.
- **Mindestgröße** — Die kleinste Anzahl von Instanzen, die an dem Standort bereitgestellt werden müssen.
- **Gewünschte Instanzen** — Die Zielanzahl der aktiven Instanzen zur Aufrechterhaltung des Standorts. Wenn aktive Instances und gewünschte Instances nicht identisch sind, wird ein Skalierungsereignis gestartet, um Instances nach Bedarf zu starten oder herunterzufahren, bis die aktiven Instances den gewünschten Instances entsprechen.
- **Maximale Größe** — Die meisten Instanzen, die am Standort bereitgestellt werden können.
- **Verfügbar** — Das Service-Limit für Instanzen abzüglich der Anzahl der verwendeten Instanzen. Dieser Wert gibt die maximale Anzahl von Instanzen an, die Sie dem Standort hinzufügen können.

Auto Scaling-Richtlinien

Dieser Abschnitt enthält Informationen zu Auto Scaling-Richtlinien, die auf die Flotte angewendet werden. Sie können eine zielbasierte Richtlinie einrichten oder aktualisieren. Die regelbasierten Richtlinien der Flotte, die mithilfe des AWS SDK oder der CLI definiert werden müssen, werden hier angezeigt. Weitere Informationen zur Skalierung finden Sie unter [Automatische Skalierung der Flottenkapazität mit Amazon GameLift Servers](#).

Verlauf der Skalierung

Sehen Sie sich Diagramme der Kapazitätsänderungen im Laufe der Zeit an.

Orte

Auf der Registerkarte Standorte sind alle Standorte aufgeführt, an denen Flotteninstanzen eingesetzt werden. Zu den Standorten gehören die Heimatregion der Flotte und alle hinzugefügten entfernten Standorte. Sie können Standorte direkt auf dieser Registerkarte hinzufügen oder entfernen.

- Standort — Name eines Standorts, an dem Flotteninstanzen bereitgestellt werden.
- Status – Hosting-Status des Flottenstandorts. Der Standortstatus verfolgt den Prozess der Aktivierung der ersten Instances am Standort. Der Standortstatus muss ACTIVE sein, um Spiele hosten zu können.
- Aktive Instances – die Anzahl der Instances mit Serverprozessen, die am Flottenstandort ausgeführt werden
- Aktive Server – die Anzahl der Spielservers-Prozesse, die Spielsitzungen am Flottenstandort hosten können
- Spielsitzungen — Die Anzahl der Spielsitzungen, die auf Instances am Standort der Flotte aktiv sind.
- Spielersitzungen — Die Anzahl der Spielersitzungen, bei denen es sich um einzelne Spieler handelt, die an Spielsitzungen teilnehmen, die am Standort der Flotte aktiv sind.

Spielsitzungen

Die Registerkarte Game sessions führt aktuelle und frühere auf der Flotte gehostete Spielsitzungen zusammen mit einigen Detailinformationen auf. Wählen Sie eine Spielsitzungs-ID, um auf zusätzliche Informationen zur Spielsitzung, einschließlich Spielersitzungen, zuzugreifen. Weitere Informationen zu Spielersitzungen finden Sie unter [Spiel- und Spielersitzungen in der Amazon GameLift Servers Konsole](#).

Spiel- und Spielersitzungen in der Amazon GameLift Servers Konsole

Sie können die Amazon GameLift Servers Konsole verwenden, um mit Spielsitzungen und Spielersitzungen zu arbeiten. Weitere Informationen zu Spielsitzungen und Spielersitzungen finden Sie unter [Amazon GameLift Servers und das Spielerlebnis](#). Die Amazon GameLift Servers Konsole bietet Informationen und Tools, mit denen Sie Probleme bei Ihren Spielsitzungen untersuchen können.

Was du tun kannst:

- Erkunden Sie die Spielsitzungen und die Aktivitäten der Spiellersitzungen, die auf einer bestimmten Flotte gehostet werden.
- Suchen Sie nach den Aktivitäten einer Spielsitzung für einen bestimmten Spieler in mehreren Flotten.
- Schließt eine bestimmte Spielsitzung.

Details zur Spielsitzung anzeigen

Die Daten zu Spielsitzungen und Spiellersitzungen werden von der Flotte organisiert, die die Spielsitzung hostet.

Um auf Informationen zur Spielsitzung und zur Spiellersitzung zuzugreifen

1. Öffnen Sie in der [Amazon GameLift ServersKonsole](#) den linken Navigationsbereich. Wählen Sie einen Hosting-Lösungstyp aus und öffnen Sie die Seite Fleets. Beispiel:
 - Hosting, überall, Flotten
 - Hosting, verwaltetes EC2, Flotten
 - Hosting, verwaltete Container, Flotten
2. Auf jeder Flottenseite wird die Liste der Flotten für Ihre aktuell ausgewählten Flotten angezeigt. AWS-Region Wähle die Flotte aus, für die du Spielsitzungsdaten einsehen möchtest.
3. Öffne auf der Detailseite der Flotte den Tab Spielsitzungen. Auf dieser Registerkarte werden alle Spielsitzungen, die auf der Flotte veranstaltet wurden, zusammen mit zusammenfassenden Informationen aufgeführt. Sie können den Tabelleninhalt mit dem Einstellungen-Tool nach Bedarf anpassen (siehe das  Symbol in der oberen rechten Ecke der Tabelle). Benutzerdefinierte Einstellungen werden für den Benutzer Ihres AWS Benutzerkontos gespeichert und automatisch angewendet, wenn Sie diese Seite aufrufen.
4. Wählen Sie eine Spielsitzung aus der Liste aus, um zusätzliche Informationen anzuzeigen.
5. Wenn die Spielsitzung Sitzungsdaten von Spielern enthält, wählen Sie Spiellersitzungen anzeigen, um das Tool zur Suche nach Spiellersitzungen zu öffnen, in dem die Spielsitzungs-ID automatisch eingetragen wird.

Die Einzelheiten der Spielsitzungen enthalten die folgenden Informationen:

- **Status** — Status der Spielsitzung.
 - **Aktivierung** — Die Instanz initiiert eine Spielsitzung.
 - **Aktiv** — Eine Spielsitzung läuft und ist für Spieler verfügbar, abhängig von den [Richtlinien zur Spielererstellung für](#) die Sitzung.
 - **Beendet** — Die Spielsitzung ist beendet.
- **ARN** — Der Amazon-Ressourcenname der Spielsitzung.
- **Name** — Name, der für die Spielsitzung generiert wurde.
- **Ort** — Der Ort, Amazon GameLift Servers an dem die Spielsitzung stattfand.
- **Erstellungszeit** — Datum und Uhrzeit der Amazon GameLift Servers Erstellung der Stream-Sitzung.
- **Endzeit** — Datum und Uhrzeit des Endes der Spielsitzung.
- **DNS-Name** — Der Hostname der Spielsitzung.
- **IP-Adresse** — Die für die Spielsitzung angegebene IP-Adresse.
- **Port** — Portnummer, die für die Verbindung zur Spielsitzung verwendet wird.
- **Creator ID** — Eine eindeutige Kennung des Spielers, der die Spielsitzung initiiert hat.
- **Richtlinie zur Erstellung von Spielersitzungen** — Gibt an, ob die Spielsitzung neue Spieler akzeptiert.
- **Richtlinie zum Schutz der Speieskalierung** — Die Art des Schutzes für Spielsitzungen, der für alle neuen Instances festgelegt werden soll, die in der Flotte Amazon GameLift Servers gestartet werden.

Spieldaten

Eigenschaftsdaten des Spiels, formatiert als Zeichenfolge, die beim Start an Ihre Spielsitzung gesendet werden sollen.

Eigenschaften des Spiels

Daten zu Spieleigenschaften, key/value paarweise formatiert, die beim Start an Ihre Spielsitzung gesendet werden.

Matchmaking-Daten

Wenn die Spielsitzung mit erstellt wurdeFlexMatch, beschreiben die Matchmaking-Daten Informationen zur Matchmaking-Konfiguration und zum Regelsatz. Dazu gehören die Spielerattribute

und Teamzuweisungen jedes Spiels. Die Daten liegen im JSON-Format vor. Weitere Informationen zum FlexMatch Matchmaking finden Sie unter [Einen Matchmaker erstellen](#).

Suchen von Spielersitzungsdaten

Wenn Ihre Game-Hosting-Lösung Spielersitzungen verwendet und einzigartige Spieler bereitstellt IDs, können Sie spieler spezifische Aktivitäten für vergangene oder aktuelle Spielsitzungen in mehreren Flotten untersuchen. Öffnen Sie das Tool zur Suche nach Spielersitzungen mithilfe einer der folgenden Methoden:

- Öffnen Sie in der Amazon GameLift Servers Konsole den linken Navigationsbereich, wählen Sie Player-Sitzungssuche und wählen Sie den zu verwendenden Filtertyp aus.
- Wenn du dir die Spielsitzungsdetails einer Flotte ansiehst, wähle Spielersitzungen anzeigen. Das Suchtool wird geöffnet. Die Spielsitzung ist bereits ausgewählt, der Filter für die Spielsitzung ist vorausgewählt und der Wert für die Spielsitzung ist eingegeben.

Wenn du das Suchtool verwendest, kannst du die folgenden Informationen angeben:

- Eine Spielersitzungs-ID, um Informationen zu einer bestimmten Spielersitzung abzurufen.
- Eine Spielsitzungs-ID, um Informationen über alle Spielersitzungen für die angeforderte Spielsitzung abzurufen. Die Ergebnisse stellen alle Spieler dar, die einen Slot reserviert oder eine Verbindung mit der Spielsitzung hergestellt haben. Sie können die Ergebnisse optional nach dem Spielersitzungsstatus filtern.
- Eine Spieler-ID, um Informationen über alle Spielersitzungen des angeforderten Spielers zu erhalten. Die Ergebnisse stellen alle Spielsitzungen dar, an denen der Spieler teilgenommen hat.

Note

Das Suchtool sucht nach allen Sitzungsaktivitäten der Spieler im aktuell ausgewählten Bereich AWS-Region. Wenn du mehrere Flotten in der Region hast, beinhalten die Ergebnisse die Sitzungsaktivitäten der Spieler in allen Flotten. Bei Flotten mit mehreren Standorten beinhalten die Ergebnisse auch die Sitzungsaktivitäten der Spieler an den entfernten Standorten der Flotten.

Es werden für jede Spiele-Sitzung die folgenden Daten zu Spielersitzungen erfasst:

- Sitzungs-ID des Spielers — Die Kennung, die der Spielsitzung zugewiesen wurde.
- Spieler-ID — Eine eindeutige Kennung für den Spieler. Wählen Sie diese ID, um zusätzliche Spielerinformationen zu erhalten.
- Spielsitzungs-ID — Die Kennung, die der Spielsitzung zugewiesen wurde.
- Flotten-ID — Die Kennung, die der Flotte zugewiesen wurde, die die Spielsitzung veranstaltet hat.
- Status — Der Status der Spielsitzung. Die folgenden Status sind möglich:
 - Reserviert — Die Spielsitzung wurde reserviert, aber der Spieler ist nicht verbunden.
 - Aktiv — Die Spielsitzung ist mit dem Spielserver verbunden.
 - Abgeschlossen — Die Spielsitzung wurde beendet; der Spieler ist nicht mehr verbunden.
 - Zeitlimit überschritten — Der Spieler konnte keine Verbindung herstellen.
- Erstellungszeit — Der Zeitpunkt, zu dem der Spieler eine Verbindung zur Spielsitzung hergestellt hat.
- Endzeit — Der Zeitpunkt, zu dem der Spieler die Verbindung zur Spielsitzung getrennt hat.
- Verbindungsdaten — IP-Adresse, DNS-Name und Port, über den der Spieler eine Verbindung zur Spielsitzung hergestellt hat.
- Spielerdaten — Informationen über den Spieler, die bei der Erstellung der Spielsitzung bereitgestellt wurden.

Beenden einer Spielsitzung

Benutze die Amazon GameLift Servers Konsole, um eine bestimmte Spielsitzung zu beenden. Diese Funktion bietet Ihnen eine einfache und schnelle Methode, um eine Spielsitzung zu lokalisieren und ein Signal zu senden, um sie zu beenden. Bei einer anderen Methode zum Beenden müssen Sie die Flotteninstanz finden, auf der die Spielsitzung läuft, remote auf die Instanz zugreifen und die Spielsitzung manuell beenden.

Du kannst eine Spielsitzung aus beliebigem Grund beenden. Der häufigste Grund ist das Auflösen einer Spielsitzung, die nicht auf natürliche Weise beendet werden kann. Infolgedessen kann die Hosting-Ressource für die Spielsitzung nicht für das Hosten einer neuen Spielsitzung freigegeben werden, und die Hosting-Kapazität der Flotte ist beeinträchtigt.

Note

Diese Funktion hängt von bestimmten Konfigurationseinstellungen für Ihre Hosting-Lösung ab. Sie hat die folgenden Einschränkungen:

- Die Spielsitzung muss auf einer Flotte gehostet werden, auf der ein Spieleserver-Build mit Server-SDK für Version Amazon GameLift Servers 5 oder höher ausgeführt wird. Wenn deine Spieleserver mit einer älteren Version bereitgestellt werden, musst du den Fernzugriff verwenden, um die Spielsitzung zu löschen.
- Wenn die Spielsitzung auf einer Anywhere-Flotte gehostet wird, muss die Flotte den Amazon GameLift Servers Agenten verwenden, um die Spielserverprozesse zu verwalten.

Um eine Spielsitzung zu beenden

1. Öffnen Sie in der [Amazon GameLift ServersKonsole](#) den linken Navigationsbereich. Wählen Sie einen Hosting-Lösungstyp aus und öffnen Sie die Seite Fleets. Beispiel:
 - Hosting, überall, Flotten
 - Hosting, verwaltetes EC2, Flotten
 - Hosting, verwaltete Container, Flotten
2. Auf jeder Flottenseite wird die Liste der Flotten für Ihre aktuell ausgewählten Flotten angezeigt. AWS-Region Wähle die Flotte aus, die die Spielsitzung hostet, die du beenden möchtest.
3. Öffne auf der Detailseite der Flotte den Tab Spielsitzungen. Wählen Sie in der Liste der Spielsitzungen die aus, die Sie beenden möchten, und klicken Sie auf die Schaltfläche Beenden.
4. In der Spielsitzung beenden? Überprüfe im Fenster, ob du die richtige Spielsitzung beendest, und wähle eine Methode zum Beenden aus.
 - Normales Herunterfahren der Spielsitzung — Diese Option sendet ein Signal an den Serverprozess, der die Spielsitzung hostet, zum Beenden. Wenn Ihr Spieleserver-Build für ordnungsgemäß integriert wurde Amazon GameLift Servers, initiiert der Serverprozess die Sequenz zum Herunterfahren der Spielsitzung, meldet, Amazon GameLift Servers dass sie beendet wird, und stoppt. Je nach Spieldesign kann die Shutdown-Sequenz Schritte beinhalten, mit denen die Spielsitzung ordnungsgemäß abgeschlossen werden kann, z. B. das Speichern von Daten und die Benachrichtigung aktiver Spieler. Bei dieser Methode ist möglicherweise eine kleine Verzögerung erforderlich, bis die Sequenz zum Herunterfahren der Spielsitzung abgeschlossen ist.
 - Sofortiges Herunterfahren der Spielsitzung — Diese Option sendet ein Signal an einen Prozessmanager, den Serverprozess, der die Spielsitzung hostet, herunterzufahren.

Diese Option umgeht das normale Herunterfahren der Spielsitzung. Es ist in der Lage, die Spielsitzung zu beenden, auch wenn der Serverprozess nicht reagieren kann.

5. Bestätigen Sie die Beendigung der Spielsitzung. Du kannst den Fortschritt des Herunterfahrens auf der Konsolenseite der Spielsitzungen verfolgen. Der Status der Spielsitzung ändert sich auf „Beendet“ und dann auf „Beendet“, wenn das Herunterfahren abgeschlossen ist.

Verwandte Themen

- Sie können Spielsitzungen auch mit dem AWS SDK und dem AWS CLI beenden. Weitere Informationen und Beispiele finden Sie im Thema Amazon GameLift Servers API-Referenz [TerminateGameSession](#).
- Weitere Informationen zur Integration von Spieleservern und zur Reaktion eines Serverprozesses auf Signale des Amazon GameLift Servers Dienstes finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Aktualisieren Sie eine Amazon GameLift Servers verwaltete Containerflotte

Sie können die meisten Eigenschaften einer verwalteten Containerflotte aktualisieren, einschließlich der Definitionen von Containergruppen. Je nachdem, welche Einstellungen aktualisiert werden, kann ein Flottenupdate eine neue Flottenbereitstellung einleiten. Bei einer Flottenbereitstellung werden alle Instanzen in der Flotte entfernt und durch Instanzen mit der neuen Konfiguration ersetzt. Zu den Einstellungen, für die eine Bereitstellung erforderlich ist, gehören:

- Definitionen von Containergruppen, einschließlich Aktualisierungen von Container-Images
- Verbindungs-Portbereiche und Berechtigungen für eingehenden Datenverkehr
- Konfiguration protokollieren

Sie können den Status von Flotteneinsätzen in der [Amazon GameLift ServersKonsole](#) oder in AWS Command Line Interface (AWS CLI) verfolgen, um eine Containerflotte zu erstellen.

Note

Sie können die Laufzeitumgebung einer Flotte nicht aktualisieren. Bei der Flottenerstellung wird das Amazon Machine Image (AMI) der Flotte auf die neueste verfügbare Version des Linux-AMI eingestellt. Alle Container-Images, die für diese Flotte bereitgestellt werden, müssen mit dieser Version kompatibel sein. Um das AMI der Flotte zu ändern oder auf

eine neuere Version zu aktualisieren, müssen Sie eine neue Flotte erstellen. Als bewährte Methode empfehlen wir, Ihre Flotten alle 30 Tage auszutauschen, um eine sichere up-to-date Laufzeitumgebung für Ihre gehosteten Spieleserver aufrechtzuerhalten. Weitere Anleitungen finden Sie unter [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#).

Console

Wähle in der [Amazon GameLift ServersKonsole](#) den AWS-Region Ort aus, an dem du die Flotte erstellen möchtest. Die Definitionen der Containergruppen müssen sich in derselben Region befinden, in der Sie die Flotte erstellen möchten.

Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie **Verwaltete Container: Flotten** aus. Wählen Sie auf der Seite „Verwaltete Containerflotten“ eine Flotte aus der Liste aus und klicken Sie auf **Bearbeiten**.

1. Aktualisieren Sie die Einstellungen für die Containerflotte nach Bedarf. Wenn Sie fertig sind, wählen Sie **Create (Erstellen)** aus.
2. Wenn für Ihre Updates eine Flottenbereitstellung erforderlich ist, werden Sie aufgefordert, die Bereitstellungsoptionen wie folgt anzugeben:
 - **Schutz von Spielsitzungen.** Sie können wählen, ob Flotteninstanzen mit aktiven Spielsitzungen geschützt werden sollen (sichere Bereitstellung). Mit dieser Einstellung werden die Flotteninstanzen erst nach dem Ende der Spielsitzungen ersetzt. Alternativ kannst du dich dafür entscheiden, Flotteninstanzen unabhängig von der Aktivität der Spielsitzung zu ersetzen (unsichere Bereitstellung). Unsichere Bereitstellungen sind während der Entwicklungs- und Testphase nützlich, um die Bereitstellungszeit zu verkürzen.
 - **Minimaler gesunder Prozentsatz.** Sie können festlegen, wie schnell die Instanzen der Flotte ersetzt werden. Verwenden Sie diese Einstellung, um während der Bereitstellung eine minimale Anzahl fehlerfreier Aufgaben aufrechtzuerhalten. Ein niedriger Wert priorisiert die Geschwindigkeit der Bereitstellung, während ein hoher Wert sicherstellt, dass die Verfügbarkeit der Spieleserver während der gesamten Bereitstellung hoch bleibt.
 - **Strategie für Fehlschläge bei der Bereitstellung.** Entscheiden Sie, welche Maßnahmen ergriffen werden sollen, wenn eine Bereitstellung fehlschlägt. Ein Bereitstellungsfehler bedeutet, dass einige der aktualisierten Container die Statusprüfungen nicht bestanden haben und daher als beeinträchtigt gelten. Sie können Bereitstellungen so einrichten, dass alle Flotteninstanzen automatisch auf den zuvor bereitgestellten Status zurückgesetzt

werden. Alternativ können Sie sich dafür entscheiden, einige der beeinträchtigten Flotteninstanzen beizubehalten, um sie beim Debuggen zu verwenden.

Wenn Ihre Anfrage erfolgreich ist, zeigt die Konsole die Registerkarte Bereitstellungen für die verwaltete Containerflotte an. Verwenden Sie diese Registerkarte, um den Status jeder Bereitstellung zu verfolgen. Wenn Sie einen neuen Einsatz für die Flotte starten, storniert diese Aktion automatisch jeden Einsatz, der derzeit für die Flotte in Bearbeitung ist.

AWS CLI

Um eine Containerflotte mit dem zu erstellen AWS CLI, öffnen Sie ein Befehlszeilenfenster und verwenden Sie den `update-container-fleet` Befehl. Weitere Informationen zu diesem Befehl finden Sie [update-container-fleet](#) in der AWS CLI Befehlsreferenz.

Im folgenden Beispiel wird eine bestehende Containerflotte mit den folgenden Merkmalen aktualisiert:

- Es aktualisiert die Definition der Spielserver-Containergruppe, sodass sie Version 2 verwendet.
- Es spezifiziert sichere Bereitstellungsoptionen.

```
{
  "DeploymentConfiguration": {
    "ImpairmentStrategy": "ROLLBACK",
    "MinimumHealthyPercentage": 75,
    "ProtectionStrategy": "WITH_PROTECTION"
  },
  "FleetId": "containerfleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "GameServerContainerGroupDefinitionName": "arn:aws:gamelift:us-west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2"
}
```

Aktualisieren Sie eine Containergruppendefinition für eine Amazon GameLift Servers Containerflotte

Sie können die meisten Eigenschaften einer Container-Gruppendefinition aktualisieren, einschließlich der einzelnen Containerdefinitionen. Container-Gruppendefinitionen haben eine Versionsnummer. Wenn Sie eine Container-Gruppendefinition aktualisieren, Amazon GameLift Servers speichert das

Update und erhöht die Versionsnummer der Definition. Bei der Konfiguration einer Containerflotte können Sie angeben, welche Version einer Containergruppendefinition bereitgestellt werden soll.

Nach der Aktualisierung einer Containergruppendefinition können Sie die neue Version für eine neue oder bestehende Containerflotte bereitstellen.

Aktualisieren Sie die Definition einer Spielserver-Containergruppe

In diesem Thema wird beschrieben, wie Sie die Definition von Spielserver-Containergruppen mithilfe der Amazon GameLift Servers Konsole oder der AWS CLI-Tools aktualisieren. Ausführlichere Informationen zu optionalen Funktionen finden Sie unter [Passen Sie eine Amazon GameLift Servers Containerflotte an](#).

So aktualisieren Sie eine Container-Gruppendefinition:

Console

Wählen Sie in der [Amazon GameLift ServersKonsole](#) den AWS-Region Ort aus, an dem Sie die Container-Gruppe erstellen möchten.

Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie **Verwaltete Container: Gruppendefinitionen**. Wählen Sie auf der Definitionsseite für Containergruppen eine Container-Gruppendefinition und -version aus, die Sie aktualisieren möchten.

Nachdem Sie Ihre Updates gespeichert haben, können Sie die neue Version verwenden, um neue Containerflotten zu erstellen, oder Sie können die Updates für eine bestehende Containerflotte bereitstellen.

Schritt 1: Definieren Sie die Details der Containergruppen-Definition

- Sie können die Einstellungen für den Gesamtspeicher und das vCPU-Limit aktualisieren.

Schritt 2: Fügen Sie Containerdefinitionen hinzu

Sie können die folgenden Aktualisierungen der Containerdefinition vornehmen:

- Aktualisieren Sie bestehende Containerdefinitionen.
- Fügen Sie neue Support-Container-Definitionen hinzu.
- Entfernen Sie Support-Container-Definitionen.

1. Sie können den ECR-Image-URI aktualisieren. Stellen Sie sicher, dass Sie die Server-SDK-Versionseinstellung so aktualisieren, dass sie mit dem neuen Image übereinstimmt.
2. Sie können den Portbereich des internen Containers nach Bedarf aktualisieren. Änderungen, die Sie an diesen Einstellungen vornehmen, können sich auf die Verbindungseinstellungen einer Containerflotte auswirken, wenn diese Änderungen in einer Flotte implementiert werden. Weitere Details finden Sie unter [Netzwerkverbindungen konfigurieren](#).

Schritt 3: Abhängigkeiten konfigurieren

- Sie können Abhängigkeiten nach Bedarf ändern. Weitere Informationen finden Sie unter [Legen Sie Container-Abhängigkeiten fest](#).

Schritt 3: Überprüfen und Erstellen

- Überprüfen Sie Ihre Aktualisierungen der Containergruppen-Definition. Verwenden Sie Bearbeiten, um weitere Änderungen in einem beliebigen Abschnitt vorzunehmen. Wenn Sie fertig sind, wählen Sie Erstellen, um eine neue Version der Container-Gruppensdefinition zu generieren.

Wenn Ihre Anfrage erfolgreich ist, zeigt die Konsole die Detailseite für die neue Container-Gruppen-Definitionsressource an. Anfänglich lautet der Status `COPYING`, als Amazon GameLift Servers beginnt, Schnappschüsse aller Container-Images für die Gruppe zu erstellen. Wenn diese Phase abgeschlossen ist, ändert sich der Status der Container-Gruppensdefinition auf `READY`. Eine Containergruppensdefinition muss den `READY` Status haben, bevor Sie damit eine Containerflotte erstellen können.

AWS CLI

Wenn Sie die AWS CLI verwenden, um eine Container-Gruppensdefinition zu erstellen oder zu aktualisieren, verwalten Sie Ihre Container-Definitions-konfigurationen in einer separaten JSON-Datei. Sie können in Ihrem CLI-Befehl auf die Datei verweisen. Schemabeispiele finden Sie unter [Erstellen Sie eine JSON Container-Definitionsdatei](#).

Wenn Sie eine Definition aktualisieren, müssen Sie nur die Werte angeben, die Sie aktualisieren möchten. Amazon GameLift Servers behält alle Werte bei, die Sie nicht in Ihre

Aktualisierungsanforderung aufnehmen. Wenn Sie eine Container-Definition ändern. Wenn Sie jedoch eine Containerdefinition ändern, geben Sie einen vollständigen Satz an.

Um eine Container-Gruppendefinition zu aktualisieren

Verwenden Sie den `update-container-group-definition` CLI-Befehl, um eine neue Container-Gruppendefinition zu aktualisieren. Weitere Informationen zu diesem Befehl finden Sie [update-container-group-definition](#) in der AWS CLI-Befehlsreferenz.

Example: Spielserver-Containergruppe

Sie können eine Version der Containergruppen-Definition angeben, wenn Sie eine Container-Gruppendefinition abrufen, aktualisieren oder löschen oder wenn Sie eine Container-Flotte erstellen oder aktualisieren. Jede Containergruppendefinition hat eine Versionseigenschaft. Darüber hinaus gibt der ARN-Wert der and-Definition die Versionsnummer an.

Dieses Beispiel veranschaulicht eine Anfrage nach einer Änderung an der Definition einer Spielserver-Containergruppe. Es wird davon ausgegangen, dass Sie eine JSON-Datei mit den Containerdefinitionen für diese Gruppe erstellt haben. In diesem Beispiel wird der ARN-Wert als Definitionsname verwendet und angegeben, dass das Update auf Version 1 erfolgt.

```
aws gamelift update-container-group-definition \
  --name arn:aws:gamelift:us-west-2:111122223333:containergroupdefinition/
MyAdventureGameContainerGroup:1 \
  --operating-system AMAZON_LINUX_2023 \
  --container-group-type GAME_SERVER \
  --total-memory-limit-mebibytes 4096 \
  --total-vcpu-limit 1 \
  --container-definitions file://SimpleServer.json
```

Klonen Sie eine Container-Gruppendefinition

Sie können die Amazon GameLift Servers Konsole verwenden, um eine bestehende Container-Gruppendefinition zu klonen.

Um eine Container-Gruppe zu klonen

1. Gehen Sie in der [Amazon GameLift ServersKonsole](#) zum linken Navigationsbereich und wählen Sie Container-Gruppen aus.

2. Wählen Sie auf der Listenseite mit den Containergruppen die vorhandene Container-Gruppe aus, die Sie klonen möchten. Nachdem Sie eine Container-Gruppe ausgewählt haben, ist die Schaltfläche Klonen aktiv.
3. Klicken auf Clone. Diese Aktion öffnet den Assistenten zum Erstellen von Containergruppen mit vordefinierten Einstellungen.
4. Geben Sie einen neuen Namen für die geklonte Containergruppe ein. Die Containergruppe in derselben Region muss eindeutige Namen haben.
5. Gehen Sie die Seiten mit der Containergruppe und der Container-Definition durch, überprüfen Sie die neue Container-Gruppe und erstellen Sie sie.

Löschen Sie eine Containergruppendefinition für eine Amazon GameLift Servers Containerflotte

Sie haben mehrere Möglichkeiten, eine Container-Gruppendefinition zu löschen. Wenn Sie eine Container-Gruppendefinition löschen, werden mit dieser Aktion auch alle Containerdefinitionen in der Container-Gruppe gelöscht.

Container-Gruppendefinitionen können mehrere Versionen haben. Container-Gruppenversionen haben denselben Namen, aber eine andere Versionsnummer. Die Container-Gruppendefinition ARNs gibt sowohl den Namen als auch die Version an.

Sie können eine Version der Containergruppen-Definition angeben, wenn Sie eine Container-Gruppendefinition abrufen, aktualisieren oder löschen oder wenn Sie eine Container-Flotte erstellen oder aktualisieren. Jede Containergruppendefinition hat eine Versionseigenschaft. Darüber hinaus gibt der ARN-Wert der Definition die Versionsnummer an.

Es gibt mehrere Möglichkeiten, Containergruppen-Definitionen zu löschen:

- Sie können alle Versionen einer bestimmten Definition löschen.
- Sie können eine bestimmte Version einer bestimmten Definition löschen.
- Sie können einige der neuesten Versionen behalten und die älteren Versionen einer bestimmten Definition löschen. Sie könnten beispielsweise alle Versionen löschen, die älter als Version 5 sind.

Sie können eine Containergruppen-Definitionsversion nur löschen, wenn sie nicht in einer Containerflotte verwendet wird.

Um eine Containergruppen-Definition zu löschen

Console

Wählen Sie in der [Amazon GameLift ServersKonsole](#) den AWS-Region Ort aus, an dem Sie die Containergruppe erstellen möchten.

Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie Verwaltete Container: Gruppenseiten. Wählen Sie auf der Seite mit den Container-Gruppenseiten die Definition aus, die Sie ändern möchten, und klicken Sie auf Löschen.

Sie werden aufgefordert, die Art des Löschvorgangs auszuwählen, den Sie vornehmen möchten, und je nach Löschtyp weitere Einstellungen festzulegen.

AWS CLI

- Um eine Container-Gruppenseite zu löschen, verwenden Sie den `delete-container-group-definition` CLI-Befehl und geben Sie Werte für die Art des Löschvorgangs an, den Sie durchführen möchten. Weitere Informationen zu diesem Befehl finden Sie [delete-container-group-definition](#) in der AWS CLI-Befehlsreferenz.

Dieses Beispiel zeigt eine Anfrage zum Löschen aller Versionen der Spielserver-Container-Gruppenseite, die älter als Version 5 sind.

Example

```
aws gamelift delete-container-group-definition \
  --name MyAdventureGameContainerGroup \
  --version-count-to-retain 5 \
```

Aliase in der Konsole Amazon GameLift Servers

Die Alias-Ressource stellt einen Zeiger auf ein Flotten- oder Terminalziel dar. Sie werden verwendet, um Spielerverkehr zu bestimmten Hosting-Zielen zu leiten. Verwenden Sie einen Alias, um ein Ziel zu abstrahieren, sodass Sie den Traffic leichter umleiten können, wenn Sie Ihre Hosting-Flotten aktualisieren.

Informationen zu Aliasnamen können Sie in der Amazon GameLift Servers Konsole oder mithilfe des oder AWS SDK für anzeigen. Amazon GameLift Servers

Console

Auf der Alias-Seite werden Informationen über Amazon GameLift Servers Aliase angezeigt, die Traffic an bestimmte Hosting-Ziele weiterleiten. Um Aliase anzuzeigen, wählen Sie im Navigationsbereich Hosting, Aliase aus.

Auf der Alias-Seite können Sie Folgendes tun:

- Erstellen Sie einen neuen Alias. Wählen Sie Alias erstellen aus.
- Filtern und sortieren Sie die Alias-Tabelle. Verwenden Sie die Steuerelemente oben in der Tabelle. Sie können den Tabelleninhalt mit dem Einstellungen-Tool nach Bedarf anpassen (siehe das  Symbol in der oberen rechten Ecke der Tabelle). Benutzerdefinierte Einstellungen werden für den Benutzer Ihres AWS Benutzerkontos gespeichert und automatisch angewendet, wenn Sie diese Seite aufrufen.
- Details zu Aliasnamen anzeigen. Wählen Sie einen Aliasnamen, um die Alias-Detailseite zu öffnen.
- Einen Alias löschen. Wählen Sie einen Alias und dann Löschen.

Aliasdetails

Auf der Seite mit den Alias-Details werden Informationen über den Alias angezeigt.

Auf dieser Seite können Sie:

- Bearbeiten Sie einen Alias. Wählen Sie Bearbeiten aus.
- Sehen Sie sich die Flotten an, die Sie mit dem Alias verknüpft haben.
- Einen Alias löschen. Wählen Sie Löschen aus.

Die Detailinformationen zu Aliasnamen schließen Folgendes ein:

- ID — Die eindeutige Nummer, die zur Identifizierung des Alias verwendet wird.
- Beschreibung — Die Beschreibung des Alias.
- ARN — Der Amazon-Ressourcenname des Alias.
- Erstellung — Datum und Uhrzeit der Erstellung des Alias.

- Letzte Aktualisierung — Datum und Uhrzeit der letzten Aktualisierung des Alias.
- Routing-Typ — Der Routing-Typ für den Alias, der einer der folgenden sein kann:
 - Einfach — Leitet Spieler-Traffic an eine angegebene Flotten-ID weiter. Sie können die Flotten-ID für einen Alias jederzeit aktualisieren.
 - Terminal — Leitet eine Nachricht zurück an den Client. Beispielsweise kannst du Spieler, die einen out-of-date Client verwenden, zu einem Ort weiterleiten, an dem sie ein Upgrade erhalten können.
- Tags — Schlüssel- und Wertepaare, die zur Identifizierung des Alias verwendet werden.

AWS SDK

Verwenden Sie die folgenden AWS CLI-Befehle, um Informationen zu dieser Ressource abzurufen:

- [ListAliases](#)
- [DescribeAlias](#)

Einen Alias bearbeiten

Sie können einen Alias bearbeiten, indem Sie Amazon GameLift Servers Konsole oder mit dem AWS CLI-Befehl [update-alias](#).

In diesem Thema wird beschrieben, wie Sie ein bearbeiten Amazon GameLift Servers Alias zur Verwendung bei der Platzierung von Spielsitzungen. Sie können die folgenden Änderungen vornehmen:

Um einen Alias zu bearbeiten

Verwenden Sie entweder Amazon GameLift Servers Konsole oder AWS Command Line Interface (AWS CLI), um einen Alias zu bearbeiten.

Console

In der [Amazon GameLift Servers Konsole](#), verwenden Sie den Navigationsbereich, um die Alias-Seite zu öffnen.

1. Wählen Sie den Alias aus, den Sie bearbeiten möchten, und klicken Sie auf Bearbeiten. Wenn Sie den Alias, den Sie bearbeiten möchten, nicht sehen, überprüfen Sie Ihren aktuell ausgewählten AWS-Region.
2. Auf der Seite Alias bearbeiten können Sie die folgenden Änderungen vornehmen:
 - Ändern Sie den Aliasnamen.
 - Ändern Sie die Alias-Beschreibung.
 - Ändern Sie die Routing-Strategie von „Einfach“ zu „Terminal“ oder von „Terminal“ zu „Einfach“.
 - Ändern Sie bei einem Alias mit einer einfachen Routing-Strategie die Flotten-ID, der der Alias zugeordnet ist.
 - Bei einem Alias mit einer Terminal-Routing-Strategie ändern Sie den Nachrichtentext.
3. Wählen Sie Änderungen speichern. Wenn Sie die Flotten-ID für einen Alias mit einer einfachen Routing-Strategie aktualisieren, kann es bis zu 2 Minuten dauern, bis die Umstellung abgeschlossen ist. Während dieser Zeit können neue Platzierungen für Spielsitzungen auf der alten Flotte stattfinden.

AWS CLI

Verwenden Sie den [update-alias](#) Befehl, um Änderungen an einer Alias-Ressource vorzunehmen. Sie können eine Aliasressource in Ihrer aktuellen Standardeinstellung AWS-Region aktualisieren oder ein `--region` Tag hinzufügen, um ein anderes anzugeben AWS-Region.

Sie können die folgenden Eigenschaften ändern:

- Aliasname.
- Alias-Beschreibung.
- Typ der Routing-Strategie. Stellen Sie sicher, dass Sie eine Flotten-ID oder eine Nachrichtenzeichenfolge für die neue Routing-Strategie angeben.
- Flotten-ID für eine bestehende einfache Routing-Strategie. Die Flotten-ID muss sich in derselben Region wie der Alias befinden.
- Nachrichtenzeichenfolge für eine bestehende Terminal-Routing-Strategie.

Warteschlangen für Spielsitzungen in der Konsole Amazon GameLift Servers

Die Warteschlangenressource stellt Ihre Konfiguration der Komponente zur Platzierung von Spielsitzungen dar. Sie bestimmt, wie Amazon GameLift Servers nach der bestmöglichen Rechenressource für das Hosten einer neuen Spielsitzung gesucht und diese ausgewählt wird, basierend auf Ihren Konfigurationseinstellungen.

Informationen zu Warteschlangen für Spielsitzungen können Sie in der Amazon GameLift Servers Konsole oder mithilfe des oder AWS SDK für Amazon GameLift Servers anzeigen.

Console

Sie können Informationen zu allen Warteschlangen einsehen, die zur Bearbeitung von Anfragen für neue Spielsitzungen verwendet werden. Auf der Seite mit den Warteschlangen werden die Warteschlangen für Spielsitzungen in den aktuell ausgewählten Spielen angezeigt. AWS-Region Auf der Seite Warteschlangen können Sie eine neue Warteschlange erstellen, vorhandene Warteschlangen löschen oder eine Detailseite für eine ausgewählte Warteschlange öffnen. Jede Seite mit den Warteschlangendetails enthält die Konfiguration und die Metrikdaten der Warteschlange. Weitere Informationen zu Warteschlangen finden Sie unter [Platzierung der Spielsitzungen konfigurieren](#).

Auf der Seite „Warteschlangen“ werden die folgenden zusammenfassenden Informationen für jede Warteschlange angezeigt. Sie können den Tabelleninhalt mit dem Einstellungen-Tool nach Bedarf anpassen (siehe das



Symbol in der oberen rechten Ecke der Tabelle). Benutzerdefinierte Einstellungen werden für den Benutzer Ihres AWS Benutzerkontos gespeichert und automatisch angewendet, wenn Sie diese Seite aufrufen.

- Warteschlangenname — Der Name, der der Warteschlange zugewiesen wurde. Dieser Name wird beispielsweise verwendet, wenn Anforderungen auf eine neue Spielsitzung eine Warteschlange referenzieren.
- Timeout in der Warteschlange — Die maximale Dauer in Sekunden, für die eine Anfrage zur Platzierung einer Spielsitzung in der Warteschlange verbleibt, bevor das Timeout eintritt.
- Ziele in der Warteschlange — Anzahl der Flotten, die in der Warteschlangenkonfiguration aufgeführt sind. Amazon GameLift Servers platziert neue Spielsitzungen auf einer beliebigen Flotte in der Warteschlange.

Details zur Warteschlange anzeigen

Sie können auf detaillierte Informationen zu einer Warteschlange zugreifen, z. B. die Konfiguration und die Metriken der Warteschlange. Um eine Seite mit Warteschlangendetails zu öffnen, gehen Sie zur Seite Warteschlangen und wählen Sie einen Warteschlangennamen aus.

Die Detailseite zu einer Warteschlange zeigt eine Übersichtstabelle und Registerkarten mit zusätzlichen Informationen an. Auf dieser Seite können Sie Folgendes machen:

- Sie können die Konfiguration der Warteschlange, die Liste der Ziele und Latenz-Richtlinien für Spieler aktualisieren. Wählen Sie Bearbeiten aus.
- Löscht eine Warteschlange. Nachdem Sie eine Warteschlange gelöscht haben, schlagen alle Anfragen für neue Spielsitzungen fehl, die auf diesen Warteschlangennamen verweisen. Wählen Sie Löschen aus.



Note

Um eine gelöschte Warteschlange wiederherzustellen, erstelle eine neue Warteschlange mit dem Namen der gelöschten Warteschlange.

Details

Im Abschnitt Übersicht werden der Amazon-Ressourcenname (ARN) der Warteschlange und das Timeout angezeigt. Sie können den ARN verwenden, wenn Sie in anderen Aktionen oder Bereichen von Amazon GameLift Servers auf die Warteschlange verweisen. Das Timeout ist die maximale Zeitspanne in Sekunden, für die eine Anfrage zur Platzierung einer Spielsitzung in der Warteschlange verbleibt, bevor das Timeout überschritten wird.

Benachrichtigung über ein Ereignis

Im Abschnitt Ereignisbenachrichtigung werden das SNS-Thema, das Ereignisbenachrichtigungen Amazon GameLift Servers veröffentlicht, und die Ereignisdaten, die zu allen von dieser Warteschlange erstellten Ereignissen hinzugefügt werden, aufgeführt.

Tags (Markierungen)

In der Tags-Tabelle werden die Schlüssel und Werte angezeigt, die zum Markieren der Ressource verwendet wurden. Weitere Informationen zum Taggen finden Sie unter Ressourcen [taggen AWS](#).

Metriken

Die Registerkarte Metrics zeigt eine grafische Darstellung der Metriken für die Warteschlange im Zeitverlauf.

Queue-Metriken umfassen eine Reihe von Informationen, die die Platzierungsaktivitäten in der Warteschlange beschreiben, einschließlich erfolgreicher Platzierungen, die nach Regionen geordnet sind. Anhand von Regionsdaten kannst du herausfinden, wo du deine Spiele veranstaltest. Metriken zur regionalen Platzierung können dabei helfen, Probleme mit dem allgemeinen Warteschlangendesign zu erkennen.

Warteschlangenmetriken sind auch bei Amazon verfügbar CloudWatch. Eine Beschreibung der verfügbaren Metriken finden Sie unter [Amazon GameLift Servers-Metriken für Warteschlangen](#).

Ziele

In der Registerkarte Destinations werden alle Ziele und Aliase für die Warteschlange angezeigt.

Wenn die Ziele Amazon GameLift Servers nach verfügbaren Ressourcen für die Ausrichtung einer neuen Spielsitzung durchsucht werden, wird die hier aufgeführte Standardreihenfolge durchsucht. Amazon GameLift Servers platziert dort neue Spielsitzungen, sofern auf dem ersten aufgelisteten Ziel noch Kapazität vorhanden ist. Sie können festlegen, dass Anfragen zur Platzierung einzelner Spielsitzungen die Standardreihenfolge überschreiben, indem Sie Daten zur Spielerlatenz angeben. Diese Daten weisen Amazon GameLift Servers an, nach einem verfügbaren Ziel mit der geringsten durchschnittlichen Spielerlatenz zu suchen. Weitere Informationen zum Entwerfen Ihrer Warteschlangen finden Sie unter [Passen Sie eine Warteschlange für Spielsitzungen an](#).

Platzierung der Sitzungen

Konfigurationseinstellungen, die den Platzierungsprozess für dein Spiel individuell anpassen.

Richtlinien für die Latenz von Spielern

Im Abschnitt Richtlinien für die Player-Latenz werden alle Richtlinien angezeigt, die von der Warteschlange verwendet werden. In der Tabelle sind die Richtlinien in der Reihenfolge aufgeführt, in der sie durchgesetzt werden.

Orte

Im Bereich Standorte werden die Orte angezeigt, an denen diese Warteschlange eine Spielsitzung platzieren kann.

Priorität

Im Bereich Priorität wird die Reihenfolge angezeigt, in der die Warteschlange die Details einer Spielsitzung auswertet.

Reihenfolge des Standorts

Im Abschnitt Reihenfolge der Standorte wird die Standardreihenfolge angezeigt, die in der Warteschlange beim Platzieren von Spielsitzungen verwendet wird. Die Warteschlange verwendet diese Reihenfolge, wenn Sie keine anderen Prioritätstypen definiert haben.

AWS SDK

Verwenden Sie die folgenden AWS CLI-Befehle, um Informationen zu dieser Ressource abzurufen:

- [DescribeGameSessionQueues](#)

Bereite dich mit Amazon GameLift Servers Hosting auf den Start vor

Verwende die folgenden Checklisten, um jede Bereitstellungsphase deines Spiels zu überprüfen. Mit [Kritisch] markierte Elemente sind entscheidend für den Produktionsstart.

Laden Sie den Fragebogen zur Amazon GameLift Servers Markteinführung herunter und füllen Sie ihn aus, der in der [Amazon GameLift ServersKonsole](#) verfügbar ist. Wir möchten, dass alle Spieleentwickler, die Spiele nutzen Amazon GameLift Servers, einen reibungslosen Starttag haben. Die angeforderten Informationen helfen uns dabei, dich auf bevorstehende Auslastungstests, Soft-Launch oder öffentliche Markteinführungen vorzubereiten. Plane ein, den ausgefüllten Fragebogen mindestens drei (3) Monate vor der Durchführung deines ersten Belastungstests einzureichen.

Themen

- [Bereite dein Spiel vor](#)
- [Bereiten Sie sich auf das Testen vor](#)
- [Bereiten Sie sich auf den Start vor](#)
- [Planen Sie Updates nach der Markteinführung ein](#)

Bereite dein Spiel vor

- [Kritisch] Vergewissern Sie sich, dass Sie alle [Schritte zur Entwicklung](#) Ihrer Hosting-Lösung abgeschlossen haben und dass Sie über alle erforderlichen Komponenten verfügen, darunter einen integrierten Spieleserver, einen Backend-Service für Spieleclients, Hosting-Flotten und eine Methode zur Platzierung von Spielsitzungen (z. B. eine Warteschlange).
- [Kritisch] [Erstellen Sie AWS Identity and Access Management \(IAM\) -Rollen](#), mit denen Ihr Spieleserver während der Ausführung auf andere AWS Ressourcen zugreifen kann.
- [Kritisch] Entwerfen und implementieren Sie nach Bedarf einen Failover auf andere Hosting-Ressourcen.
- [Plane den Rollout von Flotten an deinen Zielorten](#) und berücksichtige dabei die Warteschlange und die Flottenstruktur deines Spiels.
- [Automatisieren Sie Ihre Bereitstellung](#) mithilfe von Infrastructure as Code (IaC) mit CloudFormation und dem AWS Cloud Development Kit (AWS CDK)
- [Sammeln Sie Protokolle und Analysen](#) mithilfe von Amazon CloudWatch und Amazon Simple Storage Service (Amazon S3).

Bereiten Sie sich auf das Testen vor

- [Kritisch] [Fordern Sie eine Erhöhung der Amazon GameLift Servers Servicekontingenten](#) und anderer AWS-Service Kontingente an, damit Ihre Live-Umgebung an die Produktionsanforderungen angepasst werden kann.
- [Kritisch] Stellen Sie sicher, dass die offenen Ports auf aktiven Flotten dem Portbereich entsprechen, den Ihre Server verwenden könnten.
- [Kritisch] Schließen Sie den RDP-Port 3389 und den SSH-Port 22.
- Entwickle einen Plan für die DevOps Verwaltung deines Spiels. Wenn Sie Amazon CloudWatch Logs oder CloudWatch benutzerdefinierte Amazon-Metriken verwenden, definieren Sie Alarmer für schwerwiegende oder kritische Probleme auf der Serverflotte. Simulieren Sie Fehler und testen Sie die Runbooks.
- Stellen Sie sicher, dass die von Ihnen verwendeten Rechenressourcen die Anzahl der Serverprozesse unterstützen können, die Sie gleichzeitig auf jeder Berechnung ausführen möchten.
- [Passen Sie Ihre Skalierungsrichtlinie](#) so an, dass sie zunächst konservativer ist und mehr ungenutzte Kapazität zur Verfügung stellt, als Sie für erforderlich halten. Sie können die Kosten

später optimieren. Erwägen Sie die Verwendung einer zielbasierten Skalierungsrichtlinie mit 20 Prozent ungenutzter Kapazität.

- Verwenden Sie FlexMatch beispielsweise [Latenzregeln](#), um Spieler zuzuordnen, die sich geografisch nahe beieinander befinden. Testen Sie mit synthetischen Latenzdaten aus Ihrem Lasttest-Client, wie sich das unter Last verhält.
- Testen Sie Ihre Spielerauthentifizierung und Ihre Infrastruktur für Spielsitzungen, um festzustellen, ob sie effektiv skaliert werden kann, um der Nachfrage gerecht zu werden.
- Stellen Sie sicher, dass ein Server, der mehrere Tage lang in Betrieb war, weiterhin Verbindungen annehmen kann.
- Erhöhen Sie Ihren Support Tarif auf Business oder Enterprise, damit wir Ihnen bei Problemen oder Ausfällen weiterhelfen AWS können.

Bereiten Sie sich auf den Start vor

- [Kritisch] [Setze die Richtlinie für den Flottenschutz](#) auf vollständigen Schutz für alle aktiven Flotten, sodass aktive Spielsitzungen durch Herunterskalierung nicht unterbrochen werden.
- [Kritisch] [Stellen Sie die maximale Größe Ihrer Flotte](#) so hoch ein, dass die zu erwartende Spitzennachfrage zumindest abgedeckt werden kann. Wir empfehlen Ihnen, Ihre maximale Größe bei unerwarteter Nachfrage zu verdoppeln.
- Ermutigen Sie Ihr gesamtes Entwicklungsteam, an der Launch-Veranstaltung teilzunehmen und die Veröffentlichung Ihres Spiels in einem Launchroom zu verfolgen.
- Überwachen Sie die Latenz und das Spielererlebnis der Spieler.

Planen Sie Updates nach der Markteinführung ein

- [Passen Sie die Skalierungsrichtlinie](#) an, um die ungenutzte Kapazität auf der Grundlage der Spielerauslastung zu minimieren.
- [Ändern Sie FlexMatch die Regeln](#) oder [fügen Sie Hosting-Standorte](#) hinzu, basierend auf den Latenzdaten der Spieler und den überarbeiteten Anforderungen.
- Optimieren Sie die Laufzeitkonfiguration, um auf jeder Computerressource so viele Spielesitzungen wie möglich auszuführen. Wenn Sie die Leistungseffizienz auf diese Weise maximieren, kann sich dies direkt auf Ihre Flottenkosten auswirken, da Sie möglicherweise mehr Serverprozesse mit denselben Rechenressourcen ausführen können.

- [Verwenden Sie Ihre Analysedaten](#), um die kontinuierliche Entwicklung voranzutreiben, das Spielerlebnis und die Langlebigkeit des Spiels zu verbessern und die Monetarisierung zu optimieren.

Amazon GameLift Servers Hosting-Ressourcen verwalten mit CloudFormation

Sie können es verwenden CloudFormation , um Ihre Amazon GameLift Servers Ressourcen zu verwalten. In erstellen Sie eine Vorlage CloudFormation, die jede Ressource modelliert, und verwenden dann die Vorlage, um Ihre Ressourcen zu erstellen. Um Ressourcen zu aktualisieren, nehmen Sie die Änderungen an Ihrer Vorlage vor und verwenden CloudFormation sie, um die Aktualisierungen zu implementieren. Sie können Ihre Ressourcen in logische Gruppen, sogenannte Stacks und Stack-Sets, organisieren.

Die Verwendung CloudFormation zur Verwaltung Ihrer Amazon GameLift Servers Hosting-Ressourcen bietet eine effizientere Möglichkeit, AWS Ressourcensätze zu verwalten. Sie können die Versionskontrolle verwenden, um Vorlagenänderungen im Zeitverlauf zu verfolgen und Aktualisierungen von mehreren Teammitgliedern zu koordinieren. Sie können Vorlagen auch wiederverwenden. Wenn Sie beispielsweise ein Spiel in mehreren Regionen bereitstellen, können Sie dieselbe Vorlage verwenden, um identische Ressourcen in jeder Region zu erstellen. Sie können diese Vorlagen auch verwenden, um dieselben Ressourcensätze in einer anderen Partition bereitzustellen.

Weitere Informationen zu CloudFormation finden Sie im [AWS CloudFormation Benutzerhandbuch](#). Informationen zu Vorlagen für Amazon GameLift Servers Ressourcen finden Sie in der [Referenz zum Amazon GameLift Servers Ressourcentyp](#).

Best Practices

Eine ausführliche Anleitung zur Verwendung CloudFormation finden Sie in den [CloudFormation Best Practices](#) im AWS CloudFormation Benutzerhandbuch. Darüber hinaus haben diese bewährten Methoden eine besondere Relevanz mit Amazon GameLift Servers.

- Verwalten Sie Ihre Ressourcen konsequent durch CloudFormation. Wenn Sie Ihre Ressourcen außerhalb CloudFormation Ihrer Ressourcen ändern, werden sie nicht mehr mit Ihren Ressourcenvorlagen synchronisiert.

- Verwenden Sie CloudFormation Stacks und Stack-Sets, um mehrere Ressourcen effizient zu verwalten.
- Verwenden Sie Stacks, um Gruppen verbundener Ressourcen zu verwalten. Zum Beispiel ein Stack, der einen Build enthält, eine Flotte, die auf den Build verweist, und einen Alias, der auf die Flotte verweist. Wenn Sie Ihre Vorlage aktualisieren, um einen Build zu ersetzen, werden die mit dem Build verbundenen Flotten CloudFormation ersetzt. CloudFormation aktualisiert dann die vorhandenen Aliase so, dass sie auf die neuen Flotten verweisen. Weitere Informationen finden Sie unter [Arbeiten mit Stacks](#) im AWS CloudFormation Benutzerhandbuch.
- Verwenden Sie CloudFormation Stack-Sets, wenn Sie identische Stacks in mehreren Regionen oder AWS Konten bereitstellen. Weitere Informationen finden Sie im AWS CloudFormation Benutzerhandbuch unter [Arbeiten mit Stack-Sets](#).
- Wenn Sie Spot-Instances verwenden, schließen Sie eine On-Demand-Flotte als Alternative ein. Wir empfehlen, Ihre Vorlagen mit zwei Flotten in jeder Region, einer Flotte mit Spot-Instances und einer Flotte mit On-Demand-Instances einzurichten.
- Gruppieren Sie Ihre standortspezifischen Ressourcen und globalen Ressourcen in separaten Stacks, wenn Sie Ressourcen an mehreren Standorten verwalten.
- Platzieren Sie Ihre globalen Ressourcen in der Nähe der Dienste, die sie verwenden. Ressourcen wie Warteschlangen und Matchmaking-Konfigurationen erhalten in der Regel eine große Anzahl von Anfragen von bestimmten Quellen. Indem Sie Ihre Ressourcen in der Nähe der Quelle dieser Anfragen platzieren, minimieren Sie die Reisezeit für Anfragen und können die Gesamtleistung verbessern.
- Platzieren Sie Ihre Matchmakingkonfiguration in derselben Region wie die Spielsitzungswarteschlange, die von ihr verwendet wird.
- Erstellen Sie für jede Flotte im Stack einen separaten Alias.

Verwendung von CloudFormation Stacks

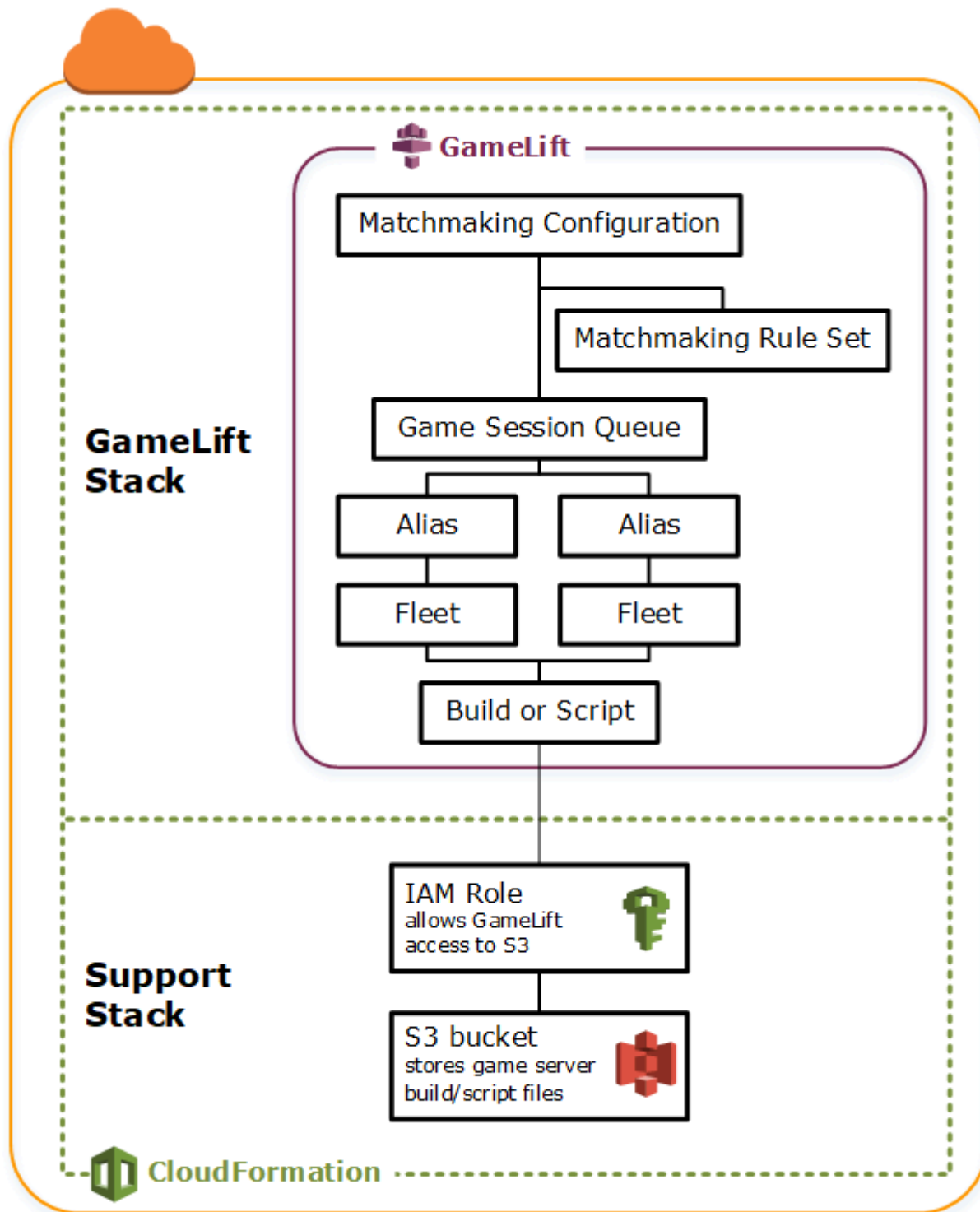
Wir empfehlen, beim Einrichten von CloudFormation Stacks für Amazon GameLift Servers Ressourcen die folgenden Strukturen zu verwenden. Deine optimale Stack-Struktur hängt davon ab, ob du dein Spiel an einem Ort oder an mehreren Orten einsetzt.

Stapel für einen einzelnen Standort

Um Amazon GameLift Servers Ressourcen an einem einzigen Standort zu verwalten, empfehlen wir eine Struktur mit zwei Stacks:

- **Support-Stack** — Dieser Stack enthält Ressourcen, von denen Ihre Amazon GameLift Servers Ressourcen abhängen. Dieser Stack sollte mindestens den S3-Bucket enthalten, in dem Sie Ihren benutzerdefinierten Spieleserver oder Echtzeit-Skriptdateien speichern. Der Stack sollte auch eine IAM-Rolle enthalten, die Ihnen die Amazon GameLift Servers Erlaubnis erteilt, Ihre Dateien aus dem S3-Bucket abzurufen, wenn Sie eine Amazon GameLift Servers Build- oder Skriptressource erstellen. Dieser Stack kann auch andere AWS Ressourcen enthalten, die mit Ihrem Spiel verwendet werden, wie DynamoDB-Tabellen, Amazon Redshift Redshift-Cluster und Lambda-Funktionen.
- **Amazon GameLift ServersStapel** — Dieser Stapel enthält all Ihre Amazon GameLift Servers Ressourcen, einschließlich des Builds oder Skripts, einer Reihe von Flotten, Aliasnamen und der Warteschlange für Spielsitzungen. CloudFormation erstellt eine Build- oder Skriptressource mit Dateien, die im S3-Bucket-Speicherort gespeichert sind, und stellt den Build oder das Skript auf einer oder mehreren Flottenressourcen bereit. Für jede Flotte sollte ein entsprechender Alias vorhanden sein. Die Spielsitzungswarteschlange verweist auf einige oder alle Flottenalias. Wenn Sie FlexMatch für das Matchmaking verwenden, enthält dieser Stapel auch eine Matchmaking-Konfiguration und einen Regelsatz.

Das folgende Diagramm zeigt eine Struktur mit zwei Stacks für die Bereitstellung von Ressourcen in einer einzigen Region. AWS



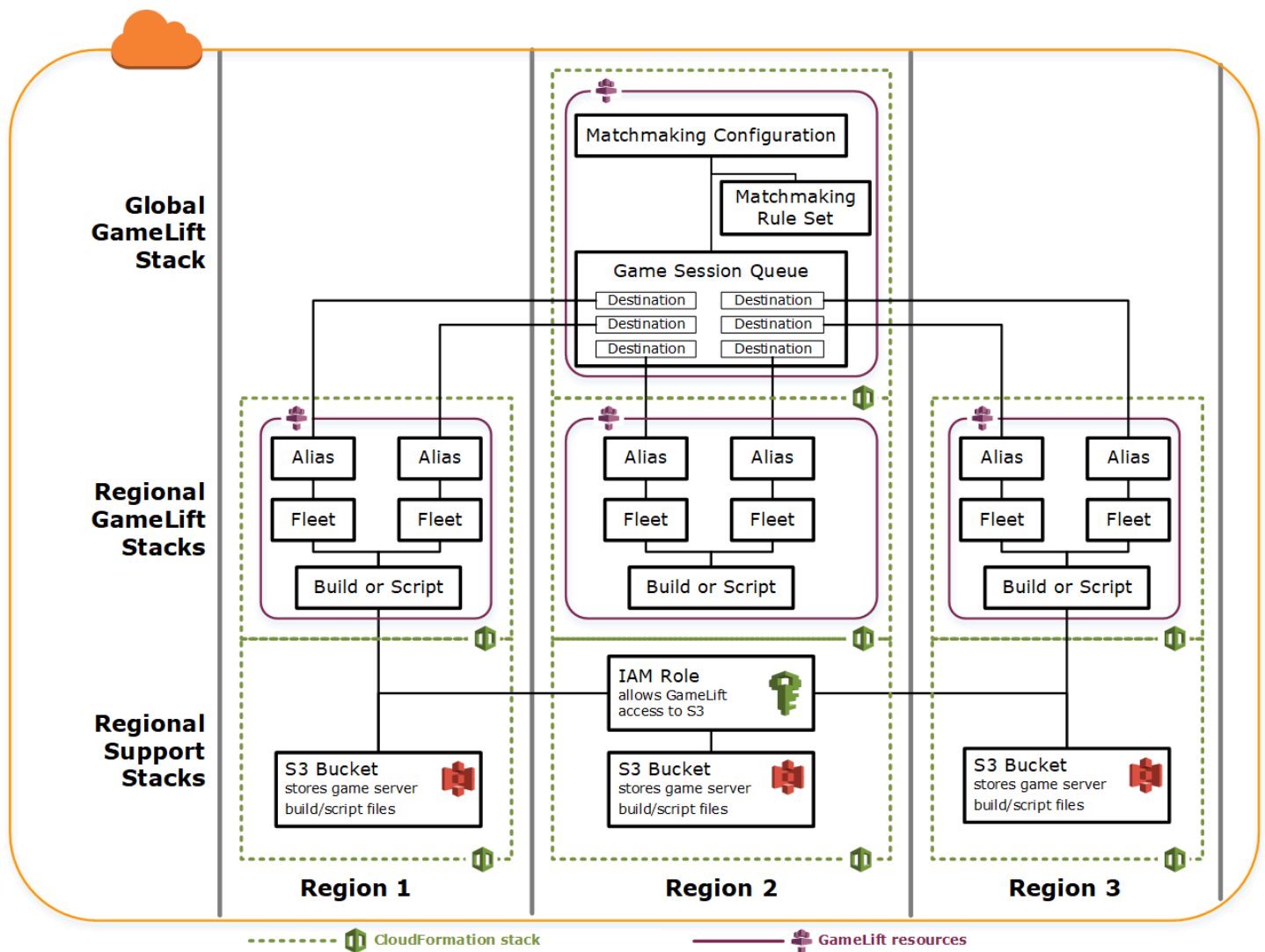
Stacks für mehrere Regionen

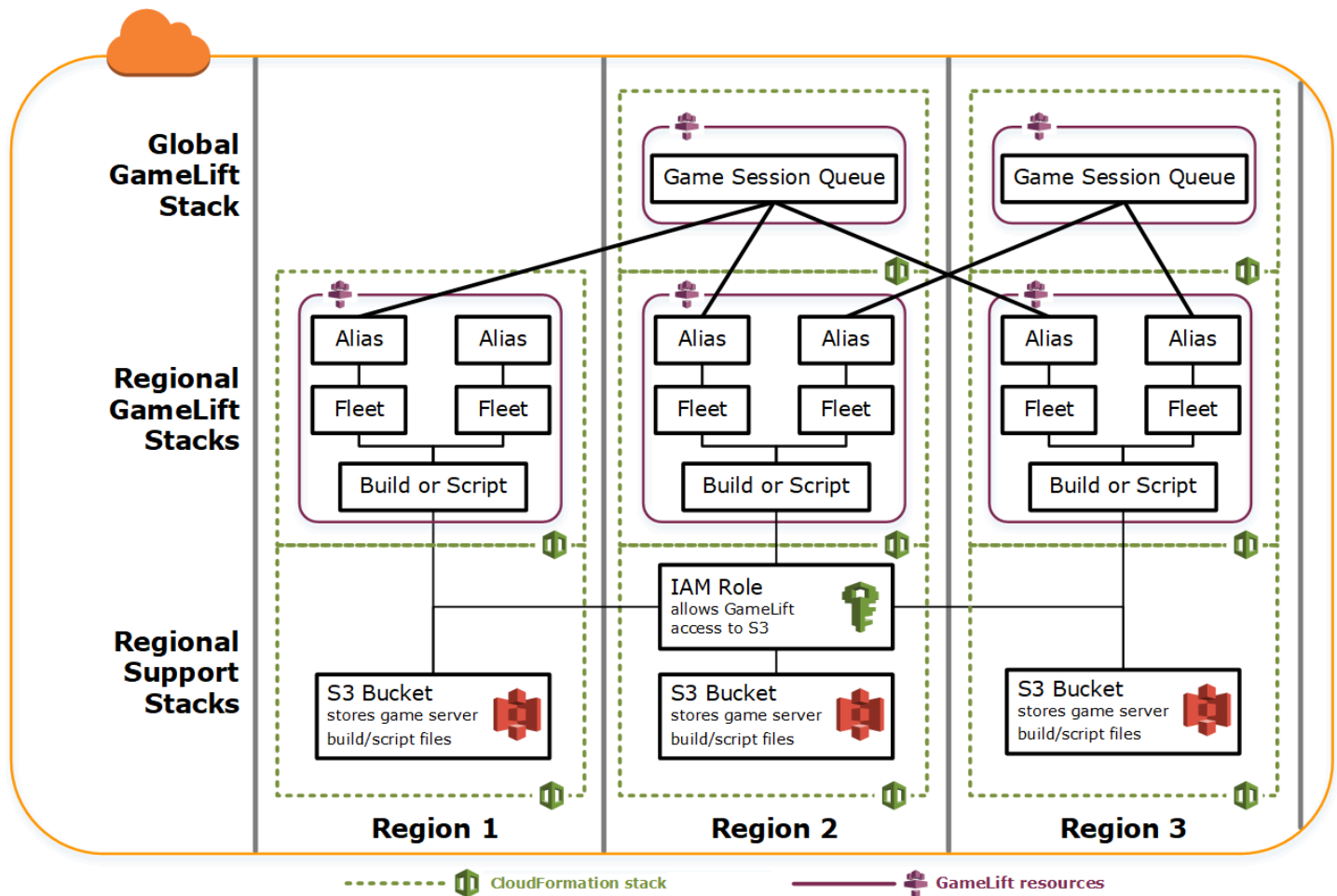
Beachten Sie bei der Bereitstellung Ihres Spiels in mehr als einer Region, wie Ressourcen in Regionen interagieren können. Einige Ressourcen, wie Amazon GameLift Servers-Flotten, können nur auf andere Ressourcen in derselben Region verweisen. Andere Ressourcen, wie z. B. eine

Amazon GameLift Servers Warteschlange, sind regionsunabhängig. Zur Verwaltung von Amazon GameLift Servers-Ressourcen in mehreren Regionen empfehlen wir die folgende Struktur.

- **Regionale Support-Stacks** — Diese Stacks enthalten Ressourcen, von denen Ihre Amazon GameLift Servers Ressourcen abhängig sind. Dieser Stack muss den S3-Bucket enthalten, in dem Sie Ihren benutzerdefinierten Spieleserver oder Echtzeit-Skriptdateien speichern. Es kann auch andere AWS Ressourcen für Ihr Spiel enthalten, wie DynamoDB-Tabellen, Amazon Redshift Redshift-Cluster und Lambda-Funktionen. Viele dieser Ressourcen sind regionsspezifisch, sodass Sie sie in jeder Region erstellen müssen. Amazon GameLift Servers benötigt außerdem eine IAM-Rolle, die den Zugriff auf diese Support-Ressourcen ermöglicht. Da eine IAM-Rolle regionsunabhängig ist, benötigen Sie nur eine Rollenressource, die sich in einer beliebigen Region befindet und auf die in allen anderen Support-Stacks verwiesen wird.
- **Regionale Amazon GameLift Servers Stacks** — Dieser Stapel enthält die Amazon GameLift Servers Ressourcen, die in jeder Region vorhanden sein müssen, in der dein Spiel eingesetzt wird, einschließlich des Build oder Skripts, einer Reihe von Flotten und Aliasnamen. CloudFormation erstellt eine Build- oder Skriptressource mit Dateien an einem S3-Bucket-Speicherort und stellt den Build oder das Skript für eine oder mehrere Flottenressourcen bereit. Für jede Flotte sollte ein entsprechender Alias vorhanden sein. Die Spielsitzungswarteschlange verweist auf einige oder alle Flottenalias. Sie können eine Vorlage verwalten, um diese Art von Stack zu beschreiben und sie verwenden, um identische Sätze von Ressourcen in jeder Region zu erstellen.
- **Globaler Amazon GameLift Servers Stapel** — Dieser Stapel enthält keine Warteschlange für Spielsitzungen und keine Matchmaking-Ressourcen. Diese Ressourcen können sich in jeder beliebigen Region befinden und werden normalerweise in derselben Region platziert. Die Warteschlange kann Flotten oder Aliase referenzieren, die sich in beliebigen Regionen befinden. Um zusätzliche Warteschlangen in verschiedenen Regionen zu platzieren, erstellen Sie zusätzliche globale Stacks.

Die folgenden Diagramme veranschaulichen eine Multistack-Struktur für den Einsatz von Ressourcen in mehreren AWS Regionen. Das erste Diagramm zeigt eine Struktur für eine einzelne Spielsitzung-Warteschlange. Das zweite Diagramm zeigt eine Struktur mit mehreren Warteschlangen.





Buids werden aktualisiert

Amazon GameLift Servers-Buids sind unveränderlich, ebenso wie die Beziehung zwischen einem Build und einer Flotte. Wenn Sie Ihre Hosting-Ressourcen aktualisieren, um einen neuen Satz von Spiele-Build-Dateien zu verwenden, müssen Sie Folgendes tun:

- Erstellen Sie einen neuen Build mit dem neuen Satz von Dateien (Ersatz).
- Erstellen Sie einen neuen Satz von Flotten, um den neuen Spiel-Build (Ersatz) bereitzustellen.
- Leiten Sie Aliase um, um auf die neuen Flotten zu verweisen (Aktualisierung ohne Unterbrechung).

Weitere Informationen finden Sie unter [Verhalten beim Aktualisieren von Stack-Ressourcen](#) im AWS CloudFormation Benutzerhandbuch.

Stellen Sie Build-Updates automatisch bereit

Bei der Aktualisierung eines Stacks, der zugehörige Build-, Flotten- und Alias-Ressourcen enthält, werden diese Schritte standardmäßig CloudFormation automatisch nacheinander ausgeführt. Sie lösen dieses Update aus, indem Sie zuerst die neuen Build-Dateien an einen neuen S3-Speicherort hochladen. Anschließend ändern Sie Ihre CloudFormation Build-Vorlage so, dass sie auf den neuen S3-Speicherort verweist. Wenn Sie Ihren Stack mit dem neuen S3-Speicherort aktualisieren, löst dies die folgende CloudFormation -Sequenz aus:

1. Ruft die neuen Dateien aus S3 ab, überprüft die Dateien und erstellt einen neuen Amazon GameLift Servers-Build.
2. Aktualisiert die Build-Referenz in der Flottenvorlage, wodurch eine neue Flottenerstellung ausgelöst wird.
3. Wenn die neuen Flotten aktiv sind, wird der Flottenverweis im Alias so aktualisiert, dass der Alias auf die neuen Flotten verweist.
4. Löscht die alte Flotte.
5. Löscht den alten Build.

Wenn Ihre Spielsitzungs-Warteschlange Flottenaliasse verwendet, wird der Spielerverkehr automatisch auf die neuen Flotten umgeschaltet, sobald die Aliase aktualisiert werden. Die alten Flotten werden schrittweise von Spielern entfernt, wenn Spielsitzungen enden. Auto Scaling übernimmt das Hinzufügen und Entfernen von Instances zu jeder Gruppe von Flotten, wenn der Spielerverkehr schwankt. Alternativ können Sie eine anfängliche gewünschte Instanzzahl angeben, um den Switch schnell hochzufahren und Auto Scaling später zu aktivieren.

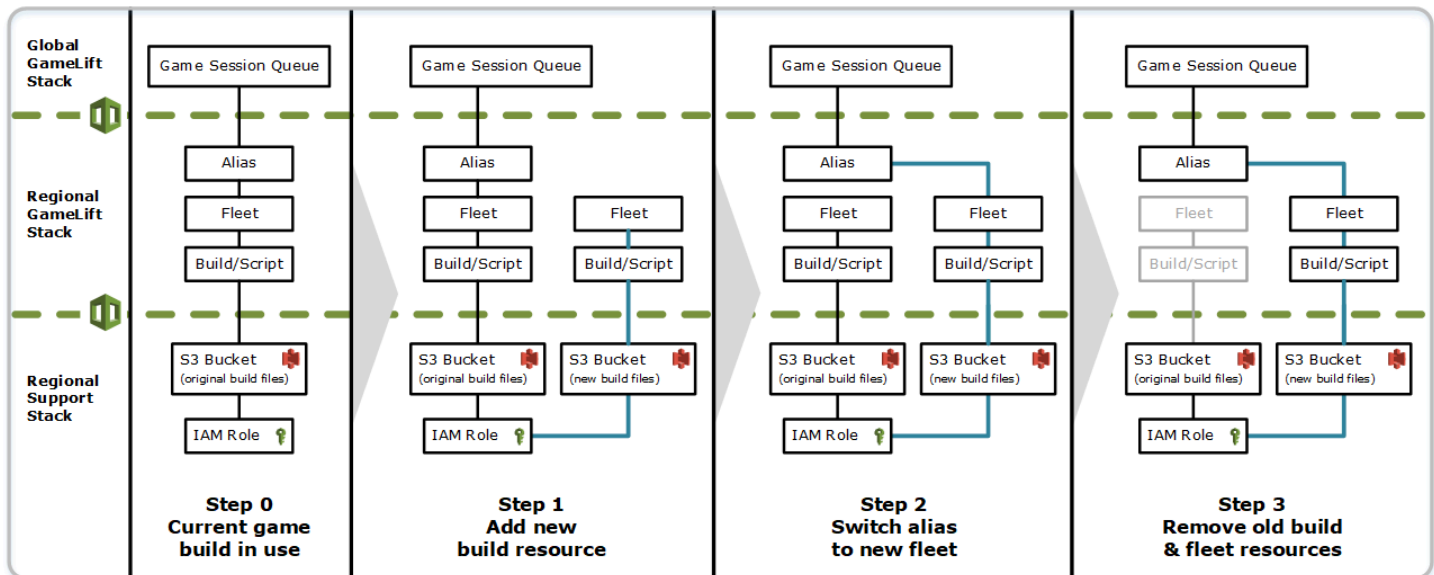
Sie können Ressourcen auch CloudFormation beibehalten lassen, anstatt sie zu löschen. Weitere Informationen finden Sie unter [RetainResources](#) in der AWS CloudFormation -API-Referenz.

Stellen Sie Build-Updates manuell bereit

Wenn Sie mehr Kontrolle darüber haben möchten, wann neue Flotten für Spieler live gehen, bieten sich Ihnen einige Möglichkeiten. Sie können Aliase manuell über die Amazon GameLift Servers-Konsole oder die CLI verwalten. Anstatt Ihre Build-Vorlage zu aktualisieren, um den Build und die Flotten zu ersetzen, können Sie auch einen zweiten Satz von Build- und Flottendefinitionen zur Vorlage hinzufügen. Wenn Sie die Vorlage aktualisieren, CloudFormation erstellt eine zweite Build-Ressource und entsprechende Flotten. Da die vorhandenen Ressourcen nicht ersetzt werden, werden sie nicht gelöscht und die Aliase verweisen weiterhin auf ursprüngliche Flotten.

Der Hauptvorteil bei diesem Ansatz ist, dass er Ihnen Flexibilität verleiht. Sie können separate Ressourcen für die neue Version Ihres Builds erstellen, die neuen Ressourcen testen und kontrollieren, wann die neuen Flotten für Spieler live gehen. Ein potenzieller Nachteil besteht darin, dass in jeder Region für einen kurzen Zeitraum doppelt so viele Ressourcen benötigt werden.

Das folgende Diagramm veranschaulicht diesen Prozess.



Wie funktionieren Rollbacks

Wenn bei einem Ressourcen-Update ein Schritt nicht erfolgreich durchgeführt wurde, löst CloudFormation automatisch ein Rollback aus. Dieser Prozess kehrt jeden Schritt in Folge um und löscht die neu erstellten Ressourcen.

Wenn Sie ein Rollback manuell auslösen müssen, setzen Sie den S3-Speicherortschlüssel der Vorlage auf den ursprünglichen Speicherort zurück und aktualisieren Sie den Stack. Ein neuer Amazon GameLift Servers-Build und eine neue Flotte werden erstellt, und der Alias wechselt zur neuen Flotte, nachdem die Flotte aktiv ist. Wenn Sie Aliase separat verwalten, müssen Sie sie so umschalten, dass sie auf die neuen Flotten verweisen.

Weitere Informationen zum Umgang mit einem fehlgeschlagenen oder hängengebliebenen Rollback finden Sie im AWS CloudFormation Benutzerhandbuch unter [Fortfahren des Rollbacks eines Updates](#).

DDoS-Schutz mit Amazon GameLift Servers

Amazon GameLift Servers Das Player-Gateway ist ein Relay-basiertes Netzwerk, das den UDP-Verkehr zwischen Spieleclients und Spieleservern weiterleitet, auf denen sie gehostet werden. Amazon GameLift Servers Player Gateway bietet proaktiven DDoS-Schutz, indem es den Datenverkehr überprüft, bevor er die Spieleserver erreicht, den Spielerverkehr begrenzt, die IP-Adressen der Spieleserver vor der Öffentlichkeit verbirgt und aktualisierte Endpunkte bereitstellt, wenn Relay-Endpunkte nicht mehr richtig funktionieren. Diese Funktion trägt dazu bei, Ihre Spieleserver vor volumetrischen Angriffen mit vernachlässigbarer zusätzlicher Latenz zu schützen.

Für Player Gateway müssen Spieleserver auf Linux-basierten Amazon GameLift Servers Managed EC2-Flotten oder Container-Flotten laufen. Du musst das Player-Gateway während der Flottenerstellung aktivieren und deinen Spielclient und dein Spiel-Backend aktualisieren, um diese Funktion nutzen zu können.

Wichtigste Vorteile

Player Gateway bietet die folgenden Sicherheits- und Betriebsvorteile:

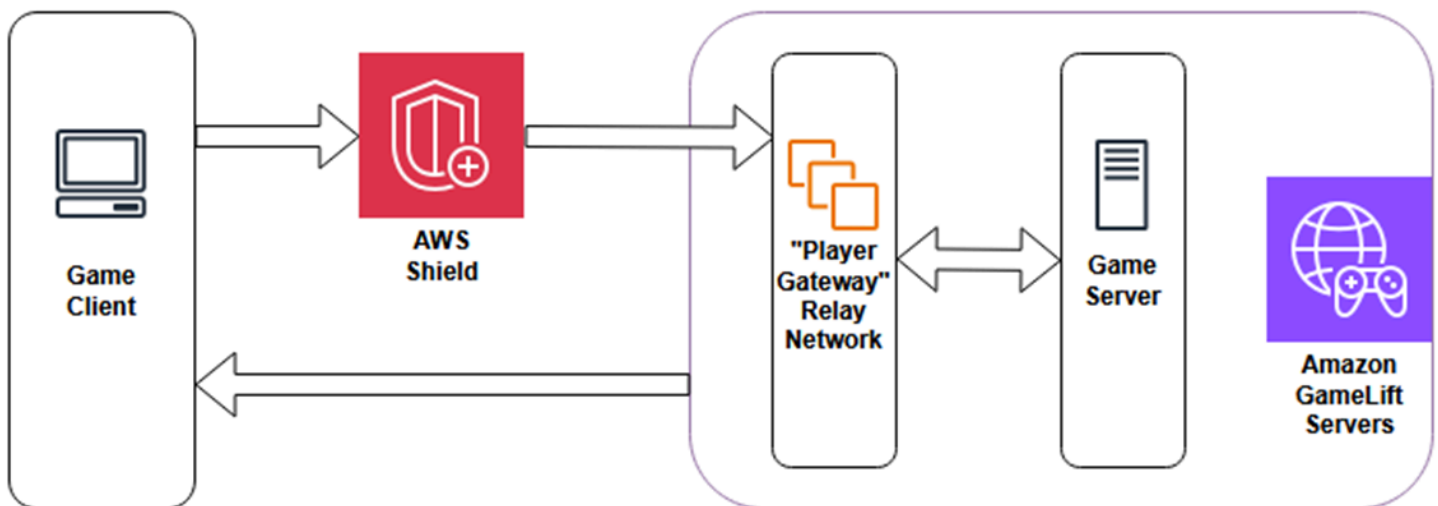
- IP-Adressen von Spieleservern verbergen — Spieleclients stellen über Relay-Endpunkte statt direkt eine Verbindung zu den Spieleservern her und verbergen so deine Spieleserveradressen vor der Öffentlichkeit.
- Datenverkehr validieren — Für den gesamten Datenverkehr über das Spieler-Gateway ist ein Spieler-Gateway-Token erforderlich, sodass nur Datenverkehr von authentifizierten Spielern deine Spieleserver erreichen kann.
- Dynamischer Austausch von Endpunkten — Wenn Relay-Endpunkte nicht mehr richtig funktionieren, werden die Endpunkte Amazon GameLift Servers schnell ersetzt und beim nächsten Anruf aktualisierte, fehlerfreie Endpunkte bereitgestellt, um die Verbindungsdetails der Spieler zu aktualisieren.
- Verteilung des Spielerverkehrs — Die Relay-Endpunkte variieren von Spieler zu Spieler, wodurch die Auswirkungen eines einzelnen fehlerhaften Relay-Endpunkts auf andere Spieler innerhalb derselben Spielsitzung reduziert werden.

So funktioniert das Player-Gateway

Das Player-Gateway verwendet ein Relay-Netzwerk, um den UDP-Verkehr zwischen Spielclients und Spieleservern weiterzuleiten. Dies schützt vor DDoS-Angriffen, indem es den Datenverkehr validiert, bevor er die Spieleserver erreicht, den Spielerverkehr begrenzt, die IP-Adressen der Spieleserver vor der Öffentlichkeit verbirgt und aktualisierte Endpunkte bereitstellt, wenn Relay-Endpunkte nicht mehr richtig funktionieren.

Datenverkehrsfluss

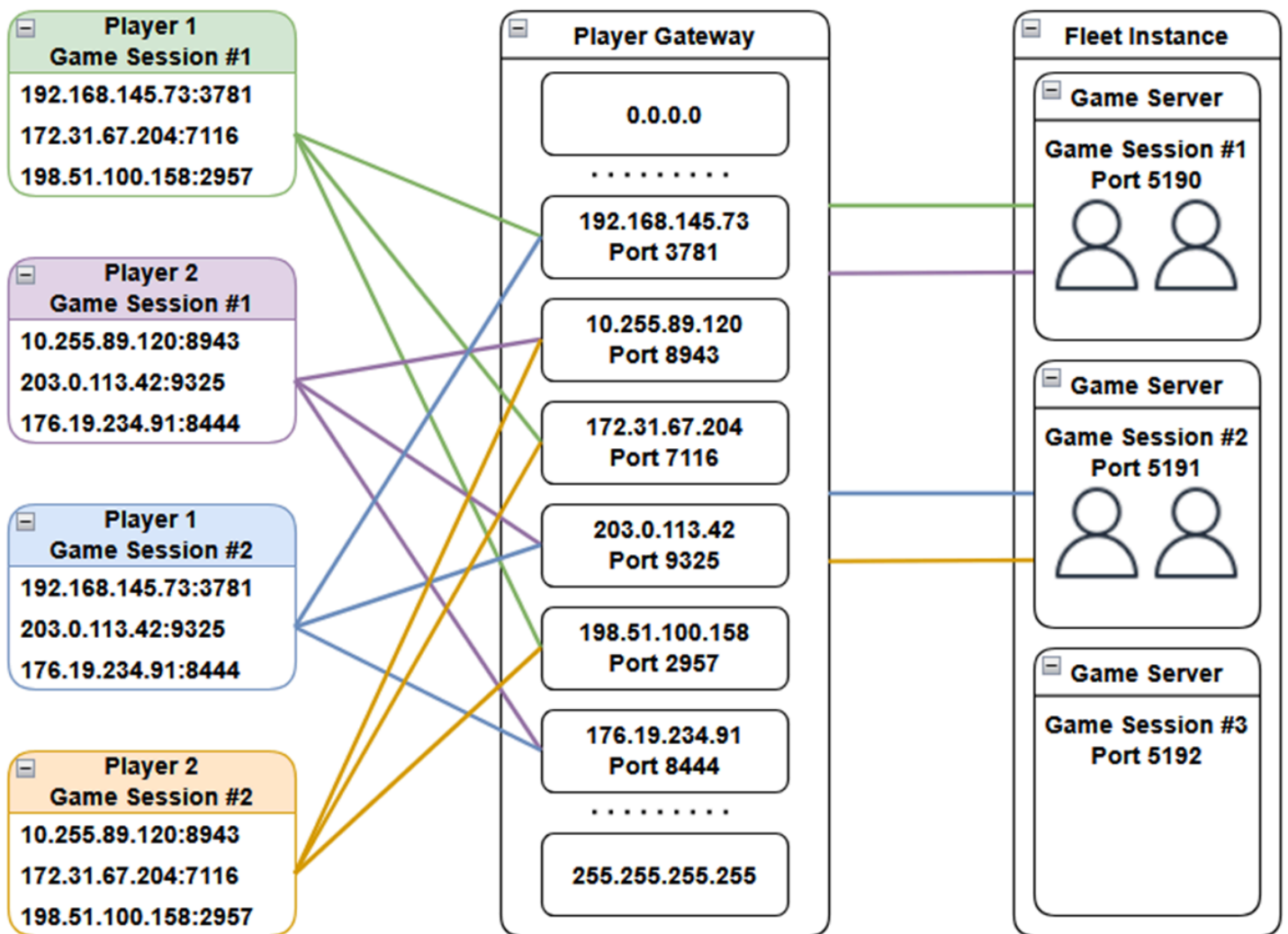
Wenn ein Spieler eine Verbindung zu einer Spielsitzung herstellt, ruft Ihr Spiel-Backend Relay-Endpunkte und ein Player-Gateway-Token von der `GetPlayerConnectionDetails` API ab und sendet sie an den Spielclient. Der Spielclient stellt das Spieler-Gateway-Token UDP-Paketen voran und sendet die Pakete an einen Relay-Endpunkt. Das Relay-Netzwerk validiert das Token und leitet legitimen Datenverkehr an den Spieleserver weiter. Vor der Auslieferung entfernt das Relay-Netzwerk das Spieler-Gateway-Token, sodass die Spieleserver die Rohdaten des Spielclients erhalten und keine Codeänderungen erforderlich sein sollten. Die Kommunikation vom Spiels server zurück zum Spielclient erfolgt unverändert über das Relay-Netzwerk.



Schlüsselkonzepte

Relay-Endpunkte

Relay-Endpunkte sind Kombinationen von IP-Adressen und Ports, die Spielclients verwenden, um Datenverkehr über das Spieler-Gateway zu senden. Jeder Spieler erhält mehrere Endpunkte, die von Spieler zu Spieler unterschiedlich sind, um den Datenverkehr zu verteilen und die Auswirkungen von Angriffen auf andere Spieler in derselben Spielsitzung zu reduzieren.



Gateway-Token für Spieler

Player-Gateway-Token sind verschlüsselte Byte-Arrays, die einen Spieler autorisieren, Traffic über das Player-Gateway an eine Spielsitzung zu senden. Die `GetPlayerConnectionDetails` API gibt Tokens als Base64-kodierte Zeichenketten zurück. Spieleclients müssen jedem UDP-Paket das Player-Gateway-Token voranstellen. Das Relay-Netzwerk validiert Token und lehnt Pakete mit ungültigen oder fehlenden Tokens ab.

⚠ Important

Spieler-Gateway-Token dürfen nicht verschlüsselt sein und müssen am Anfang jedes UDP-Pakets stehen, das vom Spielclient gesendet wird. Wenn dein Spiel Payloads verschlüsselt, stelle das unveränderte Player-Gateway-Token nach der Verschlüsselung der Spieldaten voran.

Spieler-Gateway-Token bleiben mindestens 3 Minuten gültig. Wir empfehlen, die Token alle 60 Sekunden zu aktualisieren, um sicherzustellen, dass Spieler aktualisierte Endpunkte erhalten, wenn die Relay-Endpunkte nicht mehr richtig funktionieren.

GetPlayerConnectionDetails API

Ihr Spiel-Backend ruft die `GetPlayerConnectionDetails` API auf, um Verbindungsdetails für Spieler in einer Spielsitzung abzurufen. Die API gibt entweder Relay-Endpunkte und Spieler-Gateway-Token zurück oder greift für eine direkte Verbindung auf die IP-Adresse und den Port des Spielservers zurück. Dein Spielclient sollte so konzipiert sein, dass er beide Verbindungsarten unterstützt. Rufen Sie diese API während der Spielsitzung regelmäßig auf (empfohlen alle 60 Sekunden), um aktualisierte Endpunkte zu erhalten, wenn die Relay-Endpunkte nicht mehr richtig funktionieren.

Weitere Informationen finden Sie unter [GetPlayerConnectionDetails API](#).

Überwachen des Spieler-Gateways

Player Gateway veröffentlicht Kennzahlen an Amazon, CloudWatch damit Sie die Muster des Netzwerkverkehrs überwachen, potenzielle DDoS-Angriffe identifizieren und die Relay-Leistung verfolgen können. Zu den Metriken gehören Pakete und eingehende/ausgehende Bytes, gedrosselter Datenverkehr und Spielsitzungen. Eine vollständige Liste der Player-Gateway-Metriken finden Sie unter [DDoMetriken zum S-Schutz \(Spieler-Gateway\)](#).

IPv4 und IPv6 Kompatibilität

Spielclients kommunizieren über IPv4. Das Player-Gateway wird verwendet IPv6, um mit Spieleservern zu kommunizieren. Amazon GameLift Servers übernimmt automatisch die Übersetzung zwischen IPv4 und IPv6 basierend auf Ihrer Flottenkonfiguration.

Weitere Informationen zur Konfiguration des Player-Gateways für Ihre Flotte finden Sie unter [CreateFleet](#) oder [CreateContainerFleet](#). Weitere Informationen IPv4 und IPv6 Support finden Sie unter [IPv4 und IPv6 Kompatibilität](#).

Aktiviere das Spieler-Gateway auf Flotten

Aktiviere das Spieler-Gateway, wenn du eine Flotte erstellst, um den Spieldatenverkehr über Relay-Endpunkte weiterzuleiten, die den eingehenden UDP-Verkehr zu deinen Spieleservern validieren und mit einer bestimmten Geschwindigkeit begrenzen.

Voraussetzungen

Bevor du das Player-Gateway aktivierst, überprüfe die folgenden Voraussetzungen:

- **Unterstützte Standorte** — Eine Liste der AWS Standorte, die Player Gateway unterstützen, finden Sie unter [Amazon GameLift Servers Servicestandorte](#).
- **Flottenart** — Das Player-Gateway unterstützt Linux-basierte Managed EC2-Flotten und Managed Container-Flotten.
- **Server-SDK** — Für das Player-Gateway ist Amazon GameLift Servers Server-SDK 5.0 oder höher erforderlich.

Player-Gateway-Modi

Konfigurieren Sie die Player-Gateway-Kompatibilität für Ihre Flotte mithilfe der `PlayerGatewayMode` folgenden Eigenschaft:

DEAKTIVIERT (Standard)

Fleet ist nicht mit Player Gateway kompatibel.

ENABLED (AKTIVIERT)

Fleet ist an Orten, an denen es unterstützt wird, mit Player Gateway kompatibel. Flottenstandorte, die diese Funktion nicht unterstützen, können trotzdem zur Flotte hinzugefügt werden, nutzen aber nicht das Player-Gateway, sondern direkt die IP-Adresse des Spielservers.

REQUIRED

Fleet ist mit dem Player-Gateway kompatibel. Flottenstandorte, die diese Funktion nicht unterstützen, können der Flotte nicht hinzugefügt werden.

IPv4 und IPv6 Kompatibilität

Spieleclients müssen über IPv4 mit dem Player-Gateway kommunizieren. Das Player-Gateway-Relay-Netzwerk kommuniziert über IPv6 mit Spieleservern. Verwaltete Container-Flotten übersetzen den IPv6-Verkehr automatisch in IPv4. Konfigurieren Sie für verwaltete EC2-Flotten, wie Ihre Flotte den IPv6 Verkehr verarbeitet, indem Sie die folgende Eigenschaft verwenden: `GameServerIpProtocolSupported PlayerGatewayConfiguration`

IPv4 (Standard)

Dein Spieleserver akzeptiert nur eingehenden IPv4 Traffic. Eine einfache IP-Übersetzungssoftware wird auf Ihren Instances installiert und ausgeführt, um den eingehenden IPv6 Datenverkehr zu empfangen und umzuwandeln in IPv4.

DUAL_STACK

Dein Spieleserver ist mit eingehendem Datenverkehr kompatibel. IPv6 IP-Übersetzungssoftware wird auf Ihren Instances nicht installiert oder ausgeführt. Spieleserver, die IPv6 systemeigene Unterstützung bieten, weisen möglicherweise eine etwas höhere Leistung auf.

Integrieren Sie das Player-Gateway in ein Spiel

Nachdem du eine Flotte erstellt hast, die für das Player-Gateway aktiviert ist, musst du deinen Spielclient und dein Spiel-Backend aktualisieren, um sie mit dem Player-Gateway zu integrieren. Ihr Spiel-Backend ruft Relay-Endpunkte und Spieler-Gateway-Token ab und stellt sie dann den Spielclients zur Verfügung. Spielclients senden den gesamten UDP-Verkehr an diese Relay-Endpunkte.

Die Integration folgt diesem Aufrufpfad:

1. Ihr Spiel-Backend ruft die `GetPlayerConnectionDetails` API auf, um Relay-Endpunkte und Player-Gateway-Token für jeden Spieler in einer Spielsitzung abzurufen.
2. Ihr Spiel-Backend sendet die Relay-Endpunkte und Spieler-Gateway-Token an den Spielclient.
3. Dein Spielclient stellt allen ausgehenden UDP-Paketen das Player-Gateway-Token voran und sendet die Pakete an die Relay-Endpunkte.
4. Das Relay-Netzwerk validiert das Spieler-Gateway-Token und leitet den Datenverkehr an deinen Spieleserver weiter.
5. Ihr Spieleserver sendet den Datenverkehr über dieselben Relay-Pfade zurück an Ihren Spielclient.

Backend-Integration

Ihr Spiel-Backend muss die `GetPlayerConnectionDetails` API aufrufen, um Relay-Endpunkte und Player-Gateway-Token für Spieler abzurufen. Das Backend stellt diese Informationen dann den Spielclients zur Verfügung.

GetPlayerConnectionDetails API

Die `GetPlayerConnectionDetails` API gibt Verbindungsdetails für Spieler in einer Spielsitzung zurück:

Endpunkte und Token weiterleiten

Wenn das Player-Gateway am Standort aktiviert und unterstützt wird, wird Folgendes zurückgegeben:

- Relay-Endpunkte — Mehrere Relay-Endpunkte (IP-Adresse und Port), die sich je nach Spieler unterscheiden
- Player-Gateway-Token — Token, das der Client allen UDP-Paketen voranstellen muss (gültig für mindestens 3 Minuten)
- Ablauf — Zeitstempel für den Ablauf des Player-Gateway-Tokens

Direkte Verbindung

Wenn das Player-Gateway am Standort nicht aktiviert ist oder nicht unterstützt wird, werden die IP-Adresse und der Port des Spielerservers zurückgegeben

Dein Spielclient sollte so konzipiert sein, dass er beide Verbindungsarten unterstützt.

Beispiel für einen API-Aufruf:

```
// C++ example using AWS SDK
Aws::GameLift::GameLiftClient client;
Aws::GameLift::Model::GetPlayerConnectionDetailsRequest request;
request.SetGameSessionId(gameSessionId);
request.SetPlayerIds(playerIds); // Vector of player IDs

auto outcome = client.GetPlayerConnectionDetails(request);
if (outcome.IsSuccess()) {
    auto result = outcome.GetResult();
    auto connectionDetails = result.GetPlayerConnectionDetails();

    // Process each player's connection details
    for (const auto& detail : connectionDetails) {
        std::string playerId = detail.GetPlayerId();

        // Get relay endpoints (IP address and port)
        auto endpoints = detail.GetEndpoints();
```

```
for (const auto& endpoint : endpoints) {  
    std::string ipAddress = endpoint.GetIpAddress();  
    int port = endpoint.GetPort();  
}  
  
// Get player gateway token  
auto token = detail.GetPlayerGatewayToken();  
  
// Get expiration time  
auto expiration = detail.GetExpiration();  
  
// Send endpoints and token to game client  
}  
}
```

Siehe [GetPlayerConnectionDetails](#) in der Amazon GameLift Servers API-Referenz.

Relay-Endpunkte aktualisieren

Rufen Sie `GetPlayerConnectionDetails` regelmäßig an, um Spielern aktualisierte Endpunkte zur Verfügung zu stellen, wenn die Relay-Endpunkte nicht mehr richtig funktionieren. Wir empfehlen, die API alle 60 Sekunden für alle Spieler in einer Spielsitzung aufzurufen. Um die API-Nutzung zu optimieren, können Sie mehrere Spieler IDs in einem einzigen API-Aufruf stapeln.

Important

Reguläre Refresh-Aufrufe sind der wichtigste Mechanismus, um Spielern aktualisierte Endpunkte zur Verfügung zu stellen. Spieler-Gateway-Token bleiben zwar mindestens 3 Minuten lang gültig, aber durch eine Aktualisierung alle 60 Sekunden wird sichergestellt, dass die Spieler regelmäßig aktualisierte Endpunkte erhalten.

Kundenintegration

Dein Spielclient muss allen ausgehenden UDP-Paketen Spielergateway-Token voranstellen und die Pakete an die bereitgestellten Relay-Endpunkte senden. Das Amazon GameLift Servers Client-SDK SDK for C++ und Unreal Engine bietet Hilfsprogramme, um diese Integration zu vereinfachen.

Anforderungen des Kunden

Um den Traffic über das Spieler-Gateway weiterzuleiten, muss dein Spielclient:

- **Player-Gateway-Token voranstellen** — Das Player-Gateway-Token allen ausgehenden UDP-Paketen voranstellen. Player-Gateway-Token dürfen nicht verschlüsselt sein und müssen am Anfang jedes Pakets stehen. Pakete ohne gültiges Player-Gateway-Token werden verworfen.
- **An Relay-Endpunkte senden** — Sendet alle UDP-Pakete an die bereitgestellten Relay-Endpunkte.
- **Verbindungsaktivität aufrechterhalten** — Stelle sicher, dass entweder der Spielclient ein Paket an den Spieleserver sendet oder der Spieleserver mindestens einmal alle 30 Sekunden ein Paket an den Spieler sendet. Diese Aktivität hält die Verbindung über das Relay-Netzwerk aufrecht.
- **Endpunkt-Updates durchführen** — Akzeptieren Sie aktualisierte Relay-Endpunkte und Player-Gateway-Token von Ihrem Backend (empfohlen alle 60 Sekunden) und wechseln Sie zu neuen Endpunkten, ohne die Verbindung zu unterbrechen.

Kunden-SDK

Das Amazon GameLift Servers Client-SDK SDK for C++ und Unreal Engine bietet Hilfsprogramme zur Vereinfachung der Player-Gateway-Integration:

- **Tokenverwaltung** — Fügt allen ausgehenden UDP-Paketen Player-Gateway-Token voran.
- **Endpunktauswahl** — Leitet den Datenverkehr mithilfe eines konfigurierbaren Algorithmus an Relay-Endpunkte weiter.
- **Aktualisierung der Endpunkte** — Plant regelmäßige Rückrufe, um aktualisierte Relay-Endpunkte und Player-Gateway-Token von Ihrem Backend abzurufen.

Algorithmen zur Endpunktauswahl

Das Client-SDK bietet zwei integrierte Algorithmen für die Auswahl des zu verwendenden Relay-Endpunkts:

Fallback-Algorithmus

Verwendet einen Endpunkt, bis er fehlerhaft ist, und wechselt dann zu einem anderen Endpunkt. Ideal für Menüs, Lobbys und rundenbasierte Spiele, bei denen kurze Unterbrechungen akzeptabel sind. Während des Endpunkt-Failovers können Pakete für den konfigurierten Timeout-Zeitraum (Standard: 2 Sekunden) verloren gehen.

Algorithmus für prädiktive Rotation

Durchläuft kontinuierlich alle verfügbaren Endpunkte und prognostiziert Ausfälle, bevor sie auftreten. Ideal für Echtzeit-Gameplays wie Ego-Shooter und Rennspiele, bei denen eine

konsistente Paketzustellung entscheidend ist. Erfordert, dass der Spieleserver Nachrichten mit einer konsistenten Frequenz sendet.

Sie können auch benutzerdefinierte Algorithmen implementieren, indem Sie die Basisalgorithmusklasse des SDK erweitern.

SDK-Ressourcen für Kunden

Vollständige Integrationsanweisungen und Beispielcode finden Sie in den folgenden Ressourcen:

C++

Ein Client-SDK SDK for C++ ist unter [Amazon GameLift ServersClient-SDK SDK for C++](#) verfügbar.

Unreal Engine

Ein Plugin für Unreal Engine ist im [Amazon GameLift ServersClient-SDK für Unreal Engine](#) verfügbar. Für den Zugriff ist eine Mitgliedschaft in der Epic Games-Organisation am erforderlich. GitHub Einzelheiten [findest du unter Unreal Engine GitHub auf](#).

Testen Sie Ihre Integration

Testen Sie Ihre Player-Gateway-Integration vor dem Einsatz in einer Flotte lokal mit dem Player-Gateway-Testtool. Dieses Tool simuliert die UDP-Proxyinfrastruktur des Player-Gateways und hilft dir dabei, zu überprüfen, ob dein Spiele-Client Spieler-Gateway-Token korrekt voranstellt, den Verkehr über mehrere Endpunkte leitet und mit Netzwerkverschlechterungen umgeht.

Anweisungen zur Einrichtung und Verwendung finden Sie in der [Player-Gateway-Test-App](#) unter GitHub

Best Practices

Halten Sie sich bei der Integration von Player Gateway an die folgenden bewährten Methoden:

- Endgeräte alle 60 Sekunden aktualisieren — Rufen Sie **GetPlayerConnectionDetails** alle 60 Sekunden an, um sicherzustellen, dass die Spieler regelmäßig aktualisierte Endpunkte erhalten.
- Batch-API-Aufrufe — Wenn Sie mehrere Spieler in derselben Spielsitzung aufrufen `GetPlayerConnectionDetails`, sollten Sie die Aufrufe stapeln, um den API-Overhead zu reduzieren.

- **Aufrechterhaltung der Verbindungsaktivität** — Stelle sicher, dass der Datenverkehr mindestens einmal alle 30 Sekunden zwischen dem Spielclient und dem Spieleserver fließt. Das kann entweder client-to-server oder server-to-client Verkehr sein.
- **Gehen Sie ordnungsgemäß mit Endpunkt-Updates um** — Wenn Ihr Spielclient aktualisierte Endpunkte vom Backend empfängt, wechseln Sie zu den neuen Endpunkten, ohne die Verbindung des Spielers zu unterbrechen.
- **Wähle den richtigen Algorithmus** — Wähle den Algorithmus zur Endpunktauswahl, der den Anforderungen deines Spiels am besten entspricht. Verwenden Sie den Fallback-Algorithmus für rundenbasierte Spiele und den Algorithmus für prädiktive Rotation für Echtzeitspiele.

Testen und Problembehebung Amazon GameLift Servers

Dieser Abschnitt enthält Anleitungen zur Diagnose und Lösung von Problemen mit Ihrer GameLift Amazon-Hosting-Lösung. Unabhängig davon, ob Sie Probleme bei der Entwicklung, Bereitstellung oder Produktion haben, helfen Ihnen diese Ressourcen dabei, die Grundursachen zu identifizieren und effektive Lösungen zu implementieren.

- **Entwicklungs- und Testumgebungen** — Richten Sie iterative Entwicklungsworkflows ein, die die Produktionsbedingungen widerspiegeln, sodass Sie Integrationsprobleme frühzeitig in Ihrem Entwicklungszyklus catch und lösen können.
- **Bereitstellung und Verwaltung von Flotten** — Diagnostiziere häufig auftretende Fehler bei der Flottenerstellung, Probleme bei der Aktivierung von Serverprozessen, Skalierungsprobleme und Bereitstellungsfehler, die den ordnungsgemäßen Betrieb deiner Spieleserver verhindern können.
- **Laufzeitdiagnose und Überwachung** — Greifen Sie auf Live-Flotteninstanzen zu, um in Echtzeit Fehler zu beheben, Leistungsanalysen durchzuführen und die Konfiguration zu validieren, um Probleme zu lösen, die das Spielerlebnis beeinträchtigen.

Nutzen Sie diese Ressourcen zusammen mit den GameLift Überwachungs- und Protokollierungsfunktionen von Amazon, um die optimale Leistung und Verfügbarkeit Ihrer Multiplayer-Spiele aufrechtzuerhalten.

Für iterative Entwicklung mit Amazon GameLift Servers Anywhere einrichten

Amazon GameLift Servers bietet Tools und Lösungen, mit denen Sie eine gehostete Testumgebung für die Spieleentwicklung einrichten können. Mit diesen Tools können Sie eine Umgebung schaffen, die das reale Spielerlebnis von Managed Hosting widerspiegelt Amazon GameLift Servers und einen schnellen, iterativen Entwicklungsprozess unterstützt.

Mit einer separaten Testumgebung reduzieren Sie den Aufwand für eine Amazon GameLift Servers verwaltete Flotte beim Testen. Ihr müsst nicht mehr jede neue Build-Iteration des Spielservers hochladen, eine neue Flotte dafür erstellen und dann mehr als 15 Minuten warten, bis sie aktiviert ist. Stattdessen könnt ihr einen neuen Build erstellen, die Testflotte schnell mit dem neuen Build aktualisieren, ihn starten und mit dem Testen beginnen.

Mit einer Amazon GameLift Servers Anywhere-Flotte können Sie eine Testumgebung mit einem lokalen Gerät einrichten, z. B. Ihrer Entwicklungs-Workstation. Sie können eine Testumgebung auch mithilfe einer cloudbasierten Hosting-Ressource einrichten.

Richten Sie eine Anywhere-Testumgebung ein, um eine Reihe von Szenarien zu entwickeln und zu testen, darunter:

- Testen Sie Ihre Gameserver-Integration mit dem Amazon GameLift Servers Server-SDK. Sie können auch ohne einen funktionierenden Spielclient testen, indem Sie AWS CLI-Aufrufe verwenden, um neue Spielsitzungen zu starten und Spielsitzungsergebnisse zu verfolgen.
- Testen Sie die Interaktionen zwischen Ihrem Spielclient, dem Backend-Service und dem Amazon GameLift Servers Service, während Sie Komponenten für Ihr Spiel entwickeln. Optimiere das Spielerlebnis, wenn du einem Spiel beitretest.
- Experimentieren Sie mit Ihrem FlexMatch Matchmaker-Design. Probieren Sie Variationen von Regelsätzen und andere Implementierungen von Matchmaking-Funktionen aus. Richten Sie das Matchmaking-Backfill ein und testen Sie es.
- Probiere andere Amazon GameLift Servers Hosting-Funktionen aus, wie z. B. die Runtime-Konfigurationseinstellungen (mit dem Amazon GameLift Servers Agenten) für die Lebenszyklusverwaltung von Spieleservern.
- Erstellen, testen und wiederholen Sie den Vorgang im Handumdrehen, um alle Aspekte des Spielerlebnisses Ihres Spiels, einschließlich Multiplayer-Interaktionen, in einer gehosteten Live-Umgebung zu überprüfen.

Später, wenn du dein Spiel für den Start vorbereitest, solltest du Amazon GameLift Servers verwaltete Flotten hinzufügen, um deine Hosting-Konfigurationen zu verfeinern und zusätzliche Szenarien zu testen, darunter die folgenden:

- Experimentieren und testen Sie die Warteschlangendesigns für Spielsitzungen, einschließlich der Verwendung von Flotten mit mehreren Standorten, Spot- und On-Demand-Flotten sowie mehreren Instance-Typen.
- Probieren Sie die Platzierungsoptionen für Spielsitzungen mit verwalteten Flotten aus, einschließlich der Verwendung optionaler Latenzrichtlinien und Einstellungen für die Flottenpriorisierung.
- Konfigurieren Sie die Kapazitätsskalierung mithilfe von automatischen oder manuellen Skalierungsoptionen, um der Nachfrage der Spieler gerecht zu werden.

- AWS CloudFormation Richten Sie Amazon GameLift Servers verwaltete Flotten ein, um Ihre Hosting-Ressourcen langfristig zu verwalten.

Fast Build Update Tool (nur für Entwicklungszwecke)

Bei verwalteten EC2 Flotten musst du, um ein Build-Update für den Spieleserver bereitzustellen, jeden neuen Build hochladen Amazon GameLift Servers und eine neue Flotte dafür erstellen.

Mit dem Fast Build Update Tool kannst du diese Schritte während der Entwicklung umgehen, was dir Zeit spart und eine schnellere Iteration der Entwicklung ermöglicht. Mit diesem Tool kannst du deine Spiel-Build-Dateien schnell auf allen Computern einer vorhandenen Flotte aktualisieren. Das Tool bietet mehrere Optionen: Sie können einen kompletten Spiel-Build ersetzen oder 6 bestimmte Dateien ändern, und Sie können festlegen, wie die Spielserverprozesse nach den Updates neu gestartet werden sollen. Du kannst es auch verwenden, um einzelne Computer in einer Flotte zu aktualisieren.

Um das Fast Build Update Tool herunterzuladen und mehr über seine Verwendung zu erfahren, besuchen Sie das Amazon GameLift Servers Toolkit-Repo für [das Fast Build Update Tool](#) auf Github.

Themen

- [Erstellen Sie eine Cloud-basierte Testumgebung](#)
- [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#)
- [Testen Sie Ihre Integration mit Amazon GameLift Servers Local](#)

Erstellen Sie eine Cloud-basierte Testumgebung

Note

In diesem Thema werden iterative Tests für Spiele behandelt, die in das Server-SDK für Amazon GameLift Servers Version 5.x integriert sind. Wenn Ihr Spiel das Server-SDK Version 4.x oder früher verwendet, finden Sie weitere Informationen unter. [Testen Sie Ihre Integration mit Amazon GameLift Servers Local](#)

Verwenden Sie eine Amazon GameLift Servers Anywhere-Flotte, um Ihre Spielkomponenten iterativ in einer cloudbasierten, gehosteten Umgebung zu erstellen und zu testen. Erstellen Sie eine Anywhere-Flotte mit Hosting-Ressourcen und einer Verbindung zum Amazon GameLift Servers Dienst, führen Sie Ihre Spielservers darauf aus und testen Sie die Spielfunktionen nach Bedarf.

Setze mit dem Amazon GameLift Servers Agenten eine Anywhere-Flotte ein

Wenn Ihr Gameserver-Build in Amazon GameLift Servers SDK 5.x oder höher integriert ist, können Sie ihn mit dem Amazon GameLift Servers Agenten auf einer cloudbasierten Anywhere-Flotte bereitstellen. Der Agent ist ein Hintergrundprozess, der die Lebenszyklen von Spielservern und andere Aufgaben auf jedem Computer in einer Flotte verwaltet. Zu diesen Aufgaben gehören die Registrierung des Computers bei einer Anywhere-Flotte, der Erwerb eines Authentifizierungstokens und die Ausführung von starting/stopping Spielserverprozessen auf der Grundlage einer Reihe von Anweisungen. Der Agent wird durch die Laufzeitkonfiguration einer Flotte gesteuert, die Sie während der Lebensdauer der Flotte jederzeit aktualisieren können. (Der Agent wird automatisch für verwaltete EC2-Flotten bereitgestellt.) [Weitere Informationen und die Möglichkeit, den Agenten herunterzuladen, finden Sie im Amazon GameLift Servers GitHub Repository.](#)

Richten Sie iterative Tests mit Amazon EC2 ein

Verwenden Sie den geführten Workflow in dieser [Amazon GameLift Servers Toolkit-Lösung](#), um eine cloudbasierte Hosting-Umgebung einzurichten, die das verwaltete Hosting-Erlebnis mit widerspiegelt. Amazon GameLift Servers

Das GitHub Repository bietet eine Reihe von Skripten, die die meisten Prozesse für die Einrichtung einer Testumgebung mit Amazon GameLift Servers Anywhere und dem Amazon GameLift Servers Agenten automatisieren. Es enthält auch Anleitungen zur Aktualisierung der Umgebung, wann immer Sie einen neuen Spielserver zum Testen erstellen müssen. Sie können ein einzelnes Skript ausführen, das eine Testumgebung mit einem Beispiel-Spielserver-Build bereitstellt, oder Sie können die einzelnen Schritte durchgehen, um sie mit Ihrem eigenen Gameserver-Build einzurichten.

In diesem Arbeitsablauf arbeiten Sie ausschließlich im AWS-Managementkonsole, AWS CloudShell um Skripts auszuführen und Befehlszeilenaufgaben zu erledigen.

Note

Für die Aufgaben in diesem Tutorial benötigen Sie einen AWS Kontobenzutzer mit Berechtigungen für die folgenden Dienste: Amazon GameLift Servers, AWS CloudShell, Amazon S3 AWS Systems Manager, Amazon EC2 und AWS Identity and Access

Management. Benutzer mit Administratorzugriff auf das AWS Konto verfügen bereits über die erforderlichen Berechtigungen.

Der Workflow umfasst die folgenden Aufgaben:

- Verpacke einen Spieleserver, für den gebaut wurde Amazon GameLift Servers. Der Workflow enthält ein Skript zum Erstellen eines C++-Beispielspielerservers, der bereits in das Server-SDK für Amazon GameLift Servers Version 5.x integriert wurde und bereit für das Hosting ist. Alternativ können Sie mit Ihrem eigenen Spielprojekt arbeiten, wenn Sie die Integration abgeschlossen haben.
- Richten Sie einen Amazon Simple Storage Service-Bucket ein, um Spielserver-Builds und Abhängigkeiten zu speichern. Wenn Sie neue Versionen Ihrer Spiele-Builds erstellen, können Sie diese in S3 speichern und die Skripts verwenden, um die Anywhere-Flotte für Spieltests zu aktualisieren.
- Holen Sie sich den Amazon GameLift Servers Agenten und erstellen Sie ihn. Der Agent verwaltet die Spielserverprozesse auf einer Hosting-Ressource auf der Grundlage Ihrer Konfiguration. Er verwendet dieselbe Logik und verhält sich genauso wie Amazon GameLift Servers verwaltetes EC2-Hosting.
- Richten Sie eine Anywhere-Flotte für Ihre Hosting-Ressourcen ein. Mit einer Anywhere-Flotte können Sie den Amazon GameLift Servers Service zum Hosten von Ressourcen verwenden, die nicht von verwaltet werden Amazon GameLift Servers. In diesem Schritt konfigurierst du auch die Laufzeitkonfiguration, die dem Amazon GameLift Servers Agenten mitteilt, wann und wie die Spielserverprozesse gestartet werden sollen.
- Richten Sie eine Amazon EC2 EC2-Instance ein. Dies ist Ihre Testumgebung für iteratives Testen. Es ist viel schneller, eine Standard-EC2-Instanz anstelle einer vollständig verwalteten Amazon GameLift Servers Instanz (die für den Einsatz auf Produktionsebene optimiert ist) zu verwenden. Mit einer Standard-EC2-Instance können Sie den Spieleserver bei Bedarf schnell und kontinuierlich aktualisieren.
- Stellen Sie Ihren Gameserver-Build und Ihren Amazon GameLift Servers Agenten auf der Amazon EC2 EC2-Instance bereit. Der Workflow stellt ein Skript bereit, das die neueste Version Ihres Spiel-Builds und aller Abhängigkeiten abrufen und auf Ihrer EC2-Instance installiert. In diesem Workflow gehören zu den Abhängigkeiten der Amazon GameLift Servers Agent und der CloudWatch Agent.
- Starten Sie den Amazon GameLift Servers Agenten. Nach der Installation wird der Agent automatisch gestartet und beginnt mit der Ausführung von Anweisungen. Dazu zählen:

- Registrieren Sie die EC2-Instance als Recheneinheit in der Amazon GameLift Servers Anywhere-Flotte.
- Stellen Sie eine WebSocket Verbindung mit dem Amazon GameLift Servers Dienst her und rufen Sie die neueste Laufzeitkonfiguration ab.
- Starte die Spielserversprozesse basierend auf den Anweisungen in der Laufzeitkonfiguration. In diesem Arbeitsablauf wird der Agent angewiesen, einen einzelnen Prozess der ausführbaren Datei des Spielservers zu starten.
- Testen Sie Ihre Spielszenarien. Sobald die Testumgebung eingerichtet und dein neuester Spieleserver-Build installiert ist, kannst du mit dem Testen beginnen. Der Workflow durchläuft mehrere Testschritte, einschließlich des Startens einer Spielsitzung. Greifen Sie auf die CloudWatch Spielserverprotokolle zu, um den Fortschritt zu verfolgen, während die Spielsitzung beginnt und sich darauf vorbereitet, Spieler aufzunehmen.

Bei der Entwicklung Ihrer Spielkomponenten, einschließlich eines Spielclients und eines clientseitigen Backend-Dienstes, können Sie diese in Ihre Testszenarien einbeziehen. Verwende einen Spielclient, um eine Spielsitzung anzufordern, Verbindungsinformationen vom Amazon GameLift Servers Dienst abzurufen und dann eine direkte Verbindung zur Spielsitzung herzustellen.

- Stellen Sie einen neuen Spieleserver bereit, bauen Sie ihn auf und wiederholen Sie die Tests. Während Sie Ihr Spiel entwickeln, können Sie neue Spieleserver-Builds generieren und diese dann schnell zum Testen in der EC2-Testumgebung bereitstellen. Laden Sie sie in den Amazon S3 S3-Bucket hoch und verwenden Sie dann die Workflow-Skripte, um die Testumgebung zu aktualisieren.

Stelle dein Spiel auf Amazon GameLift Servers verwaltete Flotten um

Nachdem Sie die Entwicklungstests abgeschlossen haben und bereit sind, sich auf die Markteinführung vorzubereiten, ist dies ein guter Zeitpunkt, um auf Amazon GameLift Servers verwaltete Flotten umzusteigen. Verwende verwaltete Flotten, um deine Spielhosting-Ressourcen zu optimieren und zu testen. Implementieren Sie Ihre Lösung für die Platzierung von Spielsitzungen (Warteschlangen und Matchmaker), wählen Sie die optimale Hosting-Hardware (einschließlich Spot-Flotten) und Standorte aus und wählen Sie eine Strategie zur Kapazitätsskalierung. Vielleicht möchtest du auch damit beginnen, die Lebenszyklen all deiner Spielhosting-Ressourcen, einschließlich Flotten, Warteschlangen und Matchmaker, effizienter zu verwalten. AWS CloudFormation

Der Übergang von einer cloudbasierten Anywhere-Testflotte zu einer Amazon GameLift Servers verwalteten Flotte erfordert nur minimalen Aufwand. Sie müssen keinen Spielcode ändern und können dieselben Warteschlangen und Matchmaker wiederverwenden. Führe die folgenden Aufgaben aus:

- Erstellen Sie eine Amazon GameLift Servers Build-Ressource. Bei einer Anywhere-Testflotte musst du deinen Gameserver-Build und die Abhängigkeiten manuell für jeden Flottencomputer bereitstellen. Laden Sie bei einer verwalteten Flotte Ihr Game-Build-Paket hochAmazon GameLift Servers, wodurch es automatisch auf allen Flottencomputern bereitgestellt wird. Einzelheiten [Erstelle einen Gameserver-Build für Amazon GameLift Servers](#) zum Verpacken Ihrer Spiele-Build-Dateien und zum Erstellen einer Build-Ressource mit Dateien in einem Amazon S3 S3-Bucket finden Sie unter.
- Erstellen Sie eine verwaltete Flotte. Erstellen Sie mithilfe der Konsole oder AWS CLI eine Flotte und geben Sie eine von EC2 verwaltete Flotte an. Für diese Art von Flotte sind zusätzliche Konfigurationseinstellungen erforderlich, einschließlich der Angabe der Build-Ressourcen und der Instance-Typen. Du kannst dieselbe Laufzeitkonfiguration verwenden, um den Lebenszyklus des Spielers auf jedem Flottencomputer zu verwalten. Einzelheiten [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#) zur Erstellung einer verwalteten Flotte finden Sie unter.
- Flottenalias umleiten (optional). Wenn Sie Aliase zur Verwendung mit Ihren Anywhere-Flotten einrichten, können Sie dieselben Aliase für Ihre verwalteten Flotten wiederverwenden. Einzelheiten [Erstelle eine Amazon GameLift Servers alias](#) zum Erstellen oder Aktualisieren eines Alias finden Sie unter.

Lokales Testen mit Amazon GameLift Servers Anywhere einrichten

Note

Dieses Thema behandelt lokale Tests für Spiele, die in das Server-SDK für Amazon GameLift Servers Version 5.x integriert sind. Wenn Ihr Spiel das Server-SDK Version 4.x oder früher verwendet, finden Sie weitere Informationen unter. [Testen Sie Ihre Integration mit Amazon GameLift Servers Local](#)

Verwenden Sie eine Amazon GameLift Servers Anywhere-Flotte und Ihre eigene Hardware, um Ihre Spielkomponenten iterativ in einer simulierten, gehosteten Umgebung zu erstellen und zu testen. Richten Sie eine Anywhere-Flotte ein und registrieren Sie ein lokales Gerät, um eine Verbindung

zum Amazon GameLift Servers Dienst herzustellen. Installieren Sie Ihren Gameserver-Build auf dem Gerät, starten Sie einen Gameserver-Prozess und testen Sie die Spielfunktionen nach Bedarf. Du kannst deinen Gameserver-Build so oft wie nötig aktualisieren, um jede neue Build-Iteration zu testen.

Mit einer Anywhere-Flotte können Sie mit der AWS CLI oder mit Testskripten testen. Wenn du einen Spielclient mit integriert hast Amazon GameLift Servers, kannst du den Client auf demselben lokalen Gerät oder auf einem anderen Gerät ausführen.

Lokales Testen mit einer Anywhere-Flotte ist besonders nützlich, um Ihre Gameserver-Integration mit zu testen Amazon GameLift Servers. Sie haben vollen Überblick über alle Hosting-Aktivitäten auf dem lokalen Computer sowie über Ereignisse und Protokolldaten.

Note

Verwendest du das Amazon GameLift Servers Plugin für Unreal Engine oder Unity? Zu diesen Tools gehören geführte Workflows für die Einrichtung lokaler Tests mit einer Anywhere-Flotte. Folgen Sie der Dokumentation für [Plugin für Unity: Richten Sie lokale Tests mit Amazon GameLift Servers Anywhere ein](#) oder [Plugin für Unreal: Hoste dein Spiel lokal mit Anywhere Amazon GameLift Servers](#).

Themen

- [Richten Sie eine lokale Anywhere-Flotte ein](#)
- [Aktualisiere und installiere deinen Gameserver](#)
- [Testen Sie die Aktivität der Spielsitzung](#)
- [Iteriere auf deinem Spieleserver](#)
- [Stelle dein Spiel auf Amazon GameLift Servers verwaltete Flotten um](#)

Richten Sie eine lokale Anywhere-Flotte ein

Gehen Sie wie folgt vor, um eine Anywhere-Flotte für Ihre lokale Workstation zu erstellen. Ausführliche Anweisungen zur Verwendung der AWS CLI oder der AWS-Managementkonsole for Amazon GameLift Servers finden Sie unter [Erstelle eine Amazon GameLift Servers Überall Flotte](#).

So erstellen Sie die Anywhere-Flotte

1. Erstellen Sie einen benutzerdefinierten Speicherort für Ihre lokale Workstation. (AWS CLI oder Konsole). Ein benutzerdefinierter Standort ist einfach eine Bezeichnung für die Rechenressource, die Sie in Ihre Anywhere-Flotte aufnehmen möchten. Benutzerdefinierte Ortsnamen müssen mit `beginnencustom-` beginnen. Beispiel: `custom-my_laptop`. Siehe [Erstellen Sie einen benutzerdefinierten Standort](#).
2. Erstellen Sie eine Anywhere-Flotte (AWS CLI oder Konsole). In diesem Schritt erstellen Sie die Flottenressource mit dem benutzerdefinierten Speicherort für Ihre lokale Workstation. Siehe [Erstellen Sie eine Anywhere-Flotte](#).

Notieren Sie sich die ID oder den ARN-Wert der neuen Flotte. Sie benötigen diesen Wert im nächsten Schritt.

3. Registrieren Sie Ihre lokale Workstation als Fleet Compute (nur AWS CLI). Eine Anywhere-Flotte muss über mindestens eine Rechenressource verfügen, um deine Spieleserver zu hosten. Siehe [Fügen Sie der Flotte einen Computer hinzu](#). Um der Flotte eine Rechenleistung hinzuzufügen, benötigst du die folgenden Informationen:
 - Ein Computename. Jeder Computer in einer Flotte muss einen eindeutigen Namen haben.
 - Die Anywhere-Flottenkennung. Sie können entweder das `FleetID` oder `verwendenFleetArn` verwenden.
 - Die Verbindungsinformationen des Computers. Geben Sie entweder `IpAddress` oder `DnsName` an. Auf diese Weise Amazon GameLift Servers stellen Spielclients eine Verbindung zu Spieleservern her.
 - Ein benutzerdefinierter Standort in der Anywhere-Flotte.

Notieren Sie sich den `GameLiftServiceSdkEndpoint` Rückgabewert. Du benötigst diesen Wert, wenn du deinen Spieleserver so aktualisierst, dass er auf einer Anywhere-Flotte läuft.

Aktualisiere und installiere deinen Gameserver

Bei dieser Aufgabe wird davon ausgegangen, dass Sie bereits einen Gameserver-Build in das Amazon GameLift Servers Server-SDK 5.x integriert haben. Der Integrationsprozess beinhaltet das Hinzufügen von Code zu Ihrem Spieleserver, sodass dieser mit dem Amazon GameLift Servers Dienst interagieren kann, um Spielsitzungen zu starten und zu verwalten.

Für eine Anywhere-Flotte musst du bestimmte Spielserver-Einstellungen manuell konfigurieren. Bei einer Amazon GameLift Servers verwalteten Flotte werden diese Einstellungen automatisch konfiguriert.

Um deinen Spielserver für eine Anywhere-Flotte vorzubereiten

1. Besorgen Sie sich ein Authentifizierungstoken. Ihr Spielserver muss bei jeder Kommunikation mit dem Amazon GameLift Servers Dienst ein Authentifizierungstoken enthalten. Amazon GameLift Servers Authentifizierungstoken sind kurzlebig und müssen regelmäßig aktualisiert werden.

Es hat sich bewährt, ein Skript zu erstellen, um die folgenden Aufgaben auszuführen:

- Rufen Sie die AWS CLI-Aktion `awsgamelift get-auth-token`.
- Speichern Sie den zurückgegebenen Token-Wert dort, wo ihn Spielserverprozesse abrufen können, z. B. in einer Umgebungsvariablen auf dem lokalen Computer.

Installiere das Skript zusammen mit deinem Spielserver auf dem Computer. Stellen Sie das Skript so ein, dass es ausgeführt wird, bevor Sie den ersten Spielserver-Prozess starten. Wenn die Spielserverprozesse aktiv sind, führe das Skript regelmäßig aus, um ein gültiges Authentifizierungstoken zu erhalten. Alle Spielserverprozesse auf dem Computer können dasselbe Authentifizierungstoken verwenden.

2. Aktualisiere deinen Amazon GameLift Servers Gameservercode. Als du deinen Gameservercode in das Server-SDK für integriert hast Amazon GameLift Servers, hast du der Aktion einen Aufruf hinzugefügt `InitSdk()`. Wenn der Spielserver auf einer Anywhere-Flotte läuft, erfordert dieser Aufruf zusätzliche Serverparameter. Weitere Informationen finden Sie unter [Initialisieren Sie den Serverprozess](#) und [Server-SDK 5.x für Amazon GameLift Servers](#) für Ihre Entwicklungssprache. Die Serverparameter sind:

- `webSocketUrl`— Setzen Sie diesen Parameter auf den `GameLiftServiceSdkEndpoint` Wert, der zurückgegeben wird, wenn Sie einen Computer bei der Flotte registrieren.
- `hostId`— Setzen Sie diesen Parameter auf den Computernamen, den Sie angeben, wenn Sie einen Computer bei der Anywhere-Flotte registrieren.
- `fleetId`— Setzen Sie diesen Parameter auf die ID der Anywhere-Flotte.
- `authToken`— Setzen Sie diesen Parameter auf das Token, das als Antwort auf eine Anfrage zum Abrufen eines Authentifizierungstokens für eine Berechnung zurückgegeben wird.

- `processId`— Setze diesen Parameter, um einen Spieleserverprozess zu identifizieren, der auf dem lokalen Computer läuft. Jeder gleichzeitige Spieleserverprozess muss eine eindeutige Prozess-ID haben.

Die Serverparameterwerte, die jeder Spieleserverprozess verwendet, müssen spezifisch für den Anywhere-Flottencomputer sein, auf dem der Prozess ausgeführt wird. Einzelheiten zum Abrufen der entsprechenden Werte für eine Berechnung finden Sie unter [Fügen Sie der Flotte einen Computer hinzu](#). Es hat sich bewährt, `webSocketUrl`, `hostIdFleetId`, und `authToken` als Umgebungsvariablen auf dem lokalen Computer festzulegen. Alle Serverprozesse, die auf dem Computer ausgeführt werden, verwenden diese Werte.

3. Installiere den Spielserver-Build auf dem lokalen Computer. Schließt alle Abhängigkeiten ein, die für den Betrieb des Spieleservers erforderlich sind.
4. Starte einen oder mehrere Gameserver-Prozesse, die auf dem lokalen Computer ausgeführt werden. Wenn der Spieleserver-Prozess die Server-SDK-Aktion aufruft `ProcessReady()`, ist der Prozess bereit, eine Spielsitzung zu hosten.

Testen Sie die Aktivität der Spielsitzung

Testen Sie Ihre Gameserver-Integration, indem Sie mit Spielsitzungen arbeiten. Wenn du keinen Spielclient mit integrierter Amazon GameLift Servers Funktionalität hast, kannst du die AWS CLI verwenden, um Spielsitzungen zu starten. Probiere die folgenden Szenarien aus:

- Erstellen Sie eine Spielsitzung. Rufen Sie [create-game-session](#) den Befehl (oder die [CreateGameSession](#) API-Operation) auf. Geben Sie die ID und den benutzerdefinierten Standort Ihrer Anywhere-Flotte an. Dieser Aufruf gibt eine eindeutige Kennung für die neue Spielsitzung zurück.
- Überprüfen Sie den Status der Spielsitzung. Rufen Sie [describe-game-sessions](#) den Befehl (oder die [DescribeGameSessions](#) API-Aktion) auf. Geben Sie die Spielsitzungs-ID an. Dieser Aufruf gibt detaillierte Informationen zur Spielsitzung zurück, einschließlich des Status der Spielsitzung. Spielsitzungen im Status Aktiv sind bereit, damit Spieler eine Verbindung herstellen können. Rufen Sie [list-game-sessions](#) command (oder die [ListGameSessions](#) API-Aktion) auf, um eine Liste aller Spielsitzungen für die Flotte zu erhalten.
- Connect zur Spielsitzung her. Wenn dein Spielclient die Möglichkeit hat, an einer Spielsitzung teilzunehmen, verwende die Verbindungsinformationen, die in den Informationen zur Spielsitzung enthalten sind.

Iteriere auf deinem Spieleserver

Du kannst dieselbe Anywhere Flotte und Rechenleistung verwenden, um andere Versionen deines Gameserver-Builds zu testen.

1. Säubere dein bestehendes **GameSession**. Wenn der Spieleserver-Prozess abstürzt oder nicht aufruft `ProcessEnding()`, Amazon GameLift Servers bereinigt er das, GameSession nachdem der Spieleserver aufgehört hat, Integritätsprüfungen zu senden.
2. Generieren Sie einen neuen Gameserver-Build. Nimm Änderungen an deinem Spieleserver vor und packe einen überarbeiteten Build.
3. Aktualisiere den Spielservers-Build auf deinem lokalen Computer. Ihre vorherige Anywhere-Flotte ist immer noch aktiv und Ihr Laptop ist immer noch als Rechenressource in der Flotte registriert.
4. Holen Sie sich ein aktualisiertes Autorisierungstoken. Rufen Sie den [get-compute-auth-token](#) CLI-Befehl auf und speichern Sie das Token auf dem lokalen Computer.
5. Starte einen oder mehrere Gameserver-Prozesse, die auf dem lokalen Computer ausgeführt werden. Wenn der Gameserver-Prozess aufgerufen wird `ProcessReady()`, kann er zum Testen verwendet werden.

Stelle dein Spiel auf Amazon GameLift Servers verwaltete Flotten um

Nachdem Sie die Entwicklungstests abgeschlossen haben und bereit sind, sich auf die Markteinführung vorzubereiten, ist dies ein guter Zeitpunkt, um auf Amazon GameLift Servers verwaltete Flotten umzusteigen. Verwende verwaltete Flotten, um deine Spielhosting-Ressourcen zu optimieren und zu testen. Implementieren Sie Ihre Lösung für die Platzierung von Spielsitzungen (Warteschlangen und Matchmaker), wählen Sie die optimale Hosting-Hardware (einschließlich Spot-Flotten) und Standorte aus und wählen Sie eine Strategie zur Kapazitätsskalierung. Vielleicht möchtest du auch damit beginnen, die Lebenszyklen all deiner Spielhosting-Ressourcen, einschließlich Flotten, Warteschlangen und Matchmaker, effizienter zu verwalten. AWS CloudFormation

Sie müssen einige geringfügige Änderungen vornehmen, um von einer lokalen Anywhere-Testflotte zu einer Amazon GameLift Servers verwalteten Flotte zu wechseln. Sie können dieselben Warteschlangen und Matchmaker wiederverwenden. Führen Sie die folgenden Aufgaben aus:

- Ändere den Code-Aufruf des Spieleservers auf **InitSdk()**. Entferne die Serverparameter. Verfolgt diese Informationen bei einer verwalteten Flotte Amazon GameLift Servers automatisch.

- Erstellen Sie eine Amazon GameLift Servers Build-Ressource. Bei einer Anywhere-Testflotte musst du deinen Gameserver-Build und die Abhängigkeiten manuell für jeden Flottencomputer bereitstellen. Bei einer verwalteten Flotte erstellst du dein Game-Build-Paket und lädst es hoch Amazon GameLift Servers, wodurch es automatisch auf allen Flottencomputern bereitgestellt wird. Einzelheiten [Erstelle einen Gameserver-Build für Amazon GameLift Servers](#) zum Verpacken Ihrer Spiele-Build-Dateien und zum Erstellen einer Build-Ressource mit Dateien in einem Amazon S3 S3-Bucket finden Sie unter. Fügen Sie keine Skripte hinzu, die eine Rechenleistung registrieren und ein Authentifizierungstoken abrufen, da diese Aufgaben bei verwalteten Flotten Amazon GameLift Servers automatisch erledigt werden.
- Erstellen Sie eine verwaltete Flotte. Erstellen Sie mithilfe der Konsole oder AWS CLI eine Flotte und geben Sie eine von EC2 verwaltete Flotte an. Für diese Art von Flotte sind zusätzliche Konfigurationseinstellungen erforderlich, einschließlich der Angabe der Build-Ressourcen und der Instance-Typen. Ihr müsst alle eine Laufzeitkonfiguration einrichten, um den Lebenszyklus der Spieleserver auf jedem Flottencomputer zu verwalten. Einzelheiten [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#) zur Erstellung einer verwalteten Flotte finden Sie unter.
- Flottenalias umleiten (optional). Wenn Sie Aliase zur Verwendung mit Ihren Anywhere-Flotten einrichten, können Sie dieselben Aliase für Ihre verwalteten Flotten wiederverwenden. Einzelheiten [Erstelle eine Amazon GameLift Servers alias](#) zum Erstellen oder Aktualisieren eines Alias finden Sie unter.

Testen Sie Ihre Integration mit Amazon GameLift Servers Local

Note

In diesem Thema werden nur Tests für Spiele behandelt, die in das Server-SDK für Amazon GameLift Servers Version 3.x und 4.x integriert sind. Ihr Server-SDK-Paket enthält eine kompatible Version von Amazon GameLift Servers Local. Wenn Sie das Server-SDK Version 5.x verwenden, finden Sie unter [Lokales Testen mit GameLift Server-SDK 5.x](#) Informationen zu lokalen Tests mit einer Amazon GameLift Servers Anywhere-Flotte. Anleitungen zur Migration von SDK 4.x auf 5.x finden Sie unter [Migration](#) zum Server-SDK 5.x. GameLift

Verwenden Sie Amazon GameLift Servers Local, um eine eingeschränkte Version des verwalteten Amazon GameLift Servers-Dienstes auf einem lokalen Gerät auszuführen und Ihre Spielintegration damit zu testen. Dieses Tool ist nützlich, wenn Sie die Integration Ihres Spiels als iterative Entwicklung durchführen. Die Alternative — jeden neuen Build hochzuladen Amazon GameLift

Servers und eine Flotte für dein Spiel zu konfigurieren — kann jedes Mal mehrere oder mehr Zeit in Anspruch nehmen.

Mit Amazon GameLift Servers Local können Sie Folgendes überprüfen:

- Ihr Spielserver ist korrekt in das Server-SDK integriert und kommuniziert ordnungsgemäß mit dem Amazon GameLift Servers-Service, um neue Spielsitzungen zu starten, neue Spieler zu akzeptieren und den Zustand und Status zu melden.
- Ihr Spielclient ist korrekt in das AWS SDK für integriert Amazon GameLift Servers und kann Informationen zu bestehenden Spielsitzungen abrufen, neue Spielsitzungen starten, Spieler zu Spielen hinzufügen und eine Verbindung zur Spielsitzung herstellen.

Amazon GameLift ServersLocal ist ein Befehlszeilentool, das eine eigenständige Version des verwalteten Dienstes startet. Amazon GameLift Servers Local bietet auch ein laufendes Ereignisprotokoll mit der Initialisierung von Serverprozessen, Zustandsprüfungen sowie API-Aufrufen und -Antworten. Amazon GameLift Servers Local erkennt eine Teilmenge der AWS SDK-Aktionen fürAmazon GameLift Servers. Du kannst vom AWS CLI oder von deinem Spielclient aus Anrufe tätigen. Alle API-Aktionen werden lokal genauso wie im Amazon GameLift Servers-Web-Service ausgeführt.

Jeder Serverprozess sollte nur eine einzige Spielsitzung hosten. Die Spielsitzung ist die ausführbare Datei, mit der Sie eine Verbindung zu Amazon GameLift Servers Local herstellen. Wenn die Spielsitzung abgeschlossen ist, sollten Sie den Vorgang aufrufen `GameLiftServerSDK::ProcessEnding` und dann beenden. Wenn Sie lokal mit Amazon GameLift Servers Local testen, können Sie mehrere Serverprozesse starten. Jeder Prozess stellt eine Verbindung zu Amazon GameLift Servers Local her. Sie können dann für jeden Serverprozess eine Spielsitzung erstellen. Wenn deine Spielsitzung endet, sollte dein Gameserver-Prozess beendet werden. Sie müssen dann manuell einen anderen Serverprozess starten.

Amazon GameLift Serverslocal unterstützt Folgendes APIs:

- `CreateGameSession`
- `CreatePlayerSession`
- `CreatePlayerSessions`
- `DescribeGameSessions`
- `DescribePlayerSessions`

Amazon GameLift Servers Lokal einrichten

Amazon GameLift Servers Local wird als eine ausführbare `.jar`-Datei bereitgestellt, die mit dem [Server-SDK](#) mitgeliefert wird. Sie kann unter Windows oder Linux ausgeführt und mit jeder Amazon GameLift Servers-unterstützten Sprache verwendet werden. Für die SDK-Version 4.0.2 können Sie sie von den [offiziellen GitHub Versionen](#) herunterladen. `GameLiftLocalDie.jar`-Datei ist im Artefakt `GameLift-CSharp-ServerSDK -4.0.2.zip` enthalten.

Bevor Sie Local ausführen, müssen Sie außerdem Folgendes installieren.

- Ein Build des Server-SDK für die Versionen 3.1.5 bis 4.x. Amazon GameLift Servers
- Java 8

Teste einen Spieleserver

Wenn du nur deinen Spieleserver testen möchtest, kannst du den verwenden, AWS CLI um Spielclient-Aufrufe an den Amazon GameLift Servers lokalen Dienst zu simulieren. Dies reicht aus, um zu überprüfen, ob Ihr Spiel-Server bezüglich der folgenden Punkte entsprechend Ihren Erwartungen funktioniert:

- Der Spieleserver startet ordnungsgemäß und initialisiert das Server-SDK für. Amazon GameLift Servers
- Als Teil des Startvorgangs teilt der Spielserver Amazon GameLift Servers mit, dass der Server bereit ist, Spielsitzungen zu hosten.
- Während der Ausführung sendet der Spielserver jede Minute den Systemstatus an Amazon GameLift Servers.
- Die Spiel-Server reagiert auf Anforderungen, eine neue Spielsitzung zu starten.

1. Amazon GameLift Servers Local starten.

Öffnen Sie ein Befehlszeilenfenster, navigieren Sie zu dem Verzeichnis mit der Datei **`GameLiftLocal.jar`** und führen Sie sie aus. Standardmäßig horcht Local auf Port 8080 auf Anforderungen von Spielclients. Wenn Sie eine andere Portnummer angeben möchten, verwenden Sie den Parameter `-p`. Beispiel:

```
java -jar GameLiftLocal.jar -p 9080
```

Wenn Local startet, sehen Sie Protokolle, die angeben, dass zwei lokale Server gestartet wurden, einer, der auf Ihren Spiel-Server horcht und ein zweiter, der auf Aufrufe seitens des Spielclients oder über die AWS CLI wartet. Die beiden Protokolle berichten fortlaufend weiter über die Aktivitäten auf den beiden lokalen Servern, einschließlich der Kommunikation mit Ihren Spielkomponenten.

2. Starten Sie Ihren Spiel-Server.

Starten Sie Ihren Amazon GameLift Servers-integrierten Spielserver lokal. Sie brauchen dabei nicht den Endpunkts für den Spiel-Server zu ändern.

Im Fenster der lokalen Eingabeaufforderung zeigen Protokollmeldungen an, dass Ihr Spielserver eine Verbindung mit dem Amazon GameLift Servers Local-Service hergestellt hat. Das bedeutet, dass dein Spieleserver das Server-SDK für Amazon GameLift Servers (mit) erfolgreich initialisiert hat. `InitSDK()` Das SDK hat `ProcessReady()` mit den angezeigten Protokollpfaden aufgerufen und ist nun im Erfolgsfall bereit, eine Spielsitzung zu hosten. Während der Spielserver läuft, protokolliert Amazon GameLift Servers jeden Statusbericht vom Spielserver. Das folgende Beispiel zu Protokollierungsmeldungen zeigt einen erfolgreich integrierten Spiel-Server:

```
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - SDK
connected: /127.0.0.1:64247
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - SDK pid is 17040,
sdkVersion is 3.1.5 and sdkLanguage is CSharp
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - NOTE: Only SDK
versions 3.1.5 and above are supported in GameLiftLocal!
16:50:53,451 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onProcessReady
received from: /127.0.0.1:64247 and ackRequest requested? true
16:50:53,543 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onProcessReady
data: logPathsToUpload: "C:\\game\\logs"
logPathsToUpload: "C:\\game\\error"
port: 1935

16:50:53,544 INFO || - [HostProcessManager] nioEventLoopGroup-3-1 - Registered new
process true, true,
16:50:53,558 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onReportHealth
received from /127.0.0.1:64247 with health status: healthy
```

Es können u. a. die Fehler- und Warnmeldungen angezeigt werden:

- Fehler: "ProcessReady hat keinen Prozess mit pID gefunden:<process ID>! Wurde initSDK () aufgerufen?"
- Warnung: „Der Prozessstatus für den Prozess mit der PId ist bereits vorhanden:<process ID>! Wird ProcessReady (...) mehr als einmal aufgerufen?"

3. Starte das AWS CLI.

Sobald Ihr Spiel-Server erfolgreich `ProcessReady()` ruft, können Sie die Clientaufrufe absetzen. Öffnen Sie ein Befehlszeilenfenster und starten Sie das AWS CLI -Tool. Der verwendet AWS CLI standardmäßig den Amazon GameLift Servers Webdienst-Endpunkt. Sie müssen diesen Endpunkt in allen Anfragen mit dem Parameter `--endpoint-url` mit dem lokalen Endpunkt überschreiben, wie im folgenden Beispiel dargestellt.

```
AWS gamelift describe-game-sessions --endpoint-url http://localhost:9080 --fleet-id fleet-123
```

Im AWS CLI Befehlszeilenfenster führen `AWS gamelift` Befehle zu Antworten, wie in der [AWS CLI Befehlsreferenz](#) dokumentiert.

4. Erstellen Sie eine Spielsitzung.

Reichen Sie mit dem AWS CLI eine [CreateGameSession\(\)](#) Anfrage ein. Die Anforderung sollte der erwarteten Syntax entsprechen. Bei Local kann der `FleetId`-Parameter auf eine beliebige Zeichenfolge eingestellt sein (`^fleet-\S+`).

```
AWS gamelift create-game-session --endpoint-url http://localhost:9080 --maximum-player-session-count 2 --fleet-id fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
```

Im Fenster der lokalen Eingabeaufforderung zeigen Protokollmeldungen an, dass Amazon GameLift Servers Local Ihrem Spielserver einen `onStartGameSession`-Callback gesendet hat. Wenn eine Spielsitzung erfolgreich erstellt wurde, antwortet Ihr Spiel-Server durch einen Aufruf von `ActivateGameSession`.

```
13:57:36,129 INFO || - [SDKInvokerImpl]
      Thread-2 - Finished sending event to game server to start a game session:
      arn:aws:gamelift:local::gamesession/
      fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-ab423a4b-b827-4765-
      aea2-54b3fa0818b6.
```

```

Waiting for ack response.13:57:36,143 INFO || - [SDKInvokerImpl]
Thread-2 - Received ack response: true13:57:36,144 INFO || -
[CreateGameSessionDispatcher] Thread-2 - GameSession with id:
arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-ab423a4b-b827-4765-
aea2-54b3fa0818b6
created13:57:36,227 INFO || - [SDKListenerImpl]
nioEventLoopGroup-3-1 - onGameSessionActivate received
from: /127.0.0.1:60020 and ackRequest
requested? true13:57:36,230 INFO || - [SDKListenerImpl]
nioEventLoopGroup-3-1 - onGameSessionActivate data: gameId:
"arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890"

```

Amazon GameLift Servers antwortet im AWS CLI Fenster mit einem Objekt für die Spielsitzung, einschließlich einer Spielsitzungs-ID. Beachten Sie, dass der Status der neuen Spielsitzung „Activating“ lautet. Der Status ändert sich zu Aktiv, sobald Ihr Spiele-Server aufgerufen wird. `ActivateGameSession` Wenn du den geänderten Status sehen möchtest, verwende den, AWS CLI um anzurufen `DescribeGameSessions()`.

```

{
  "GameSession": {
    "Status": "ACTIVATING",
    "MaximumPlayerSessionCount": 2,
    "FleetId": "fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d",
    "GameSessionId": "arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890",
    "IpAddress": "127.0.0.1",
    "Port": 1935
  }
}

```

Teste einen Spiele-Server und Client

Um die gesamte Integration Ihres Spiels zu überprüfen (inklusive der Verbindung von Spielern mit Spielen), können Sie gleichzeitig Ihren Spiel-Server und Ihren Spieleclient lokal ausführen. Dadurch können Sie programmgesteuerte Aufrufe von Ihrem Spielclient an Amazon GameLift Servers Local testen. Sie können die folgenden Aktionen überprüfen:

- Der Spielclient sendet erfolgreich AWS SDK-Anfragen an den Amazon GameLift Servers lokalen Dienst, unter anderem um Spielsitzungen zu erstellen, Informationen über bestehende Spielsitzungen abzurufen und Spielersitzungen zu erstellen.
- Die Spiel-Server validiert die Spieler ordnungsgemäß, wenn diese versuchen, einem Spiel beizutreten. Bei validierte Spielern kann der Spiel-Server Daten über den Spieler abrufen (falls implementiert).
- Die Spiel-Server meldet den Verbindungsverlust, wenn ein Spieler das Spiel verlässt.
- Die Spiel-Server meldet das Ende einer Spielsitzung.

1. Amazon GameLift Servers Local starten.

Öffnen Sie ein Befehlszeilenfenster, navigieren Sie zu dem Verzeichnis mit der Datei *GameLiftLocal.jar* und führen Sie sie aus. Standardmäßig horcht Local auf Port 8080 auf Anforderungen von Spielclients. Wenn Sie eine andere Portnummer angeben möchten, verwenden Sie den Parameter `-p`. Beispiel:

```
./gamelift-local -p 9080
```

Wenn Local startet, sehen Sie Protokolle, die angeben, dass zwei lokale Server gestartet wurden, einer, der auf Ihren Spiel-Server horcht und ein zweiter, der auf Aufrufe seitens des Spielclients oder über die AWS CLI wartet.

2. Starten Sie Ihren Spiel-Server.

Starten Sie Ihren Amazon GameLift Servers-integrierten Spielserver lokal. Weitere Informationen zu Meldungsprotokollen finden Sie unter [Teste einen Spieleserver](#).

3. Konfigurieren Sie Ihren Spieleclient für Local und starten Sie ihn.

Um Ihren Spielclient mit dem Amazon GameLift Servers Local-Service zu verwenden, müssen Sie an der Einrichtung Ihres Spielclients die folgenden Änderungen vornehmen, wie unter [Richten Sie die Amazon GameLift Servers API ein](#) beschrieben:

- Ändern Sie das `ClientConfiguration`-Objekt so, dass es auf Ihren lokalen Endpunkt verweist, z. B. `http://localhost:9080`.
- Legen Sie einen Wert für die Zielflotten-ID fest. Bei Local brauchen Sie keine echte Flotten-ID anzugeben, es reicht aus, eine beliebige gültige Zeichenfolge (`^fleet-\S+`) anzugeben, z. B. `fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d`.

- AWS Anmeldeinformationen einrichten. Für Lokale benötigen Sie keine echten AWS - Anmeldeinformationen. Sie können den Zugriffsschlüssel und den geheimen Schlüssel auf eine beliebige Zeichenfolge festlegen.

Sobald Sie den Spielclient gestartet haben, sollten die Protokollmeldungen im lokalen Eingabeaufforderungsfenster anzeigen, dass er den `GameLiftClient` initialisiert hat und erfolgreich mit dem Amazon GameLift Servers-Service kommuniziert wird.

4. Spielclient-Aufrufe an den Amazon GameLift Servers-Service testen.

Überprüfen Sie, ob Ihr Spieleclient erfolgreich einige oder alle der folgenden API-Aufrufe ausführen kann:

- [`CreateGameSession\(\)`](#)
- [`DescribeGameSessions\(\)`](#)
- [`CreatePlayerSession\(\)`](#)
- [`CreatePlayerSessions\(\)`](#)
- [`DescribePlayerSessions\(\)`](#)

In dem Local-Befehlszeilenfenster werden nur Aufrufe an `CreateGameSession()` mit entsprechenden Meldungen protokolliert. Protokollmeldungen zeigen an, wenn Amazon GameLift Servers Local Ihren Spielserver auffordert, eine Spielsitzung zu starten (`onStartGameSession`-Callback) und einen erfolgreichen `ActivateGameSession` erhält, wenn Ihr Spielserver ihn aufruft. In dem AWS CLI -Fenster führen alle API-Aufrufe entweder zu Antworten oder zu Fehlermeldungen, wie dokumentiert.

5. Überprüfen Sie, ob Ihr Spiel-Server die Verbindungen neuer Spieler validiert.

Stellen Sie nach dem Erstellen einer Spielsitzung und einer Spielersitzung eine direkte Verbindung zu der Spielsitzung her.

In dem Befehlszeilenfenster von Local sollten Meldungen darauf hinweisen, dass der Spiel-Server eine `AcceptPlayerSession()`-Anfrage gesendet hat, um die neue Spielerverbindung zu überprüfen. Wenn Sie das Anrufen verwenden `DescribePlayerSessions()`, sollte AWS CLI sich der Sitzungsstatus des Spielers von Reserviert auf Aktiv ändern.

6. Überprüfen Sie, ob Ihr Spielserver dem Amazon GameLift Servers-Service den Spiel- und Spielerstatus meldet.

Damit Amazon GameLift Servers die Spielernachfrage verwaltet und Metriken korrekt gemeldet werden können, muss Ihr Spielserver verschiedene Status an Amazon GameLift Servers zurückmelden. Überprüfen Sie, ob Local Ereignisse zu den folgenden Aktionen protokolliert. Möglicherweise möchten Sie den auch verwenden, AWS CLI um Statusänderungen zu verfolgen.

- Spieler trennt sich von einer Spielsitzung — Amazon GameLift Servers Lokale Protokollnachrichten sollten zeigen, dass dein Spielserver anruft `RemovePlayerSession()`. Ein AWS CLI -Aufruf an `DescribePlayerSessions()` sollte eine Statusänderung von `Active` nach `Completed` sichtbar machen. Sie können auch `DescribeGameSessions()` aufrufen, um zu überprüfen, ob sich die Anzahl der Spieler in der aktuellen Spielsitzung um 1 vermindert.
- Spielsitzung endet — Amazon GameLift Servers Lokale Protokollnachrichten sollten zeigen, dass dein Spielserver anruft `TerminateGameSession()`.

 Note

Bisher wurde empfohlen, `TerminateGameSession()` beim Beenden einer Spielsitzung anzurufen. Diese Methode ist mit Amazon GameLift Servers Server SDK v4.0.1 veraltet. Siehe [Beenden Sie eine Spielsitzung](#).

- Der Serverprozess ist beendet — Amazon GameLift Servers Lokale Protokollnachrichten sollten zeigen, dass dein Spielserver anruft. `ProcessEnding()` Ein AWS CLI Aufruf von `DescribeGameSessions()` sollte eine Statusänderung von `Active` zu `Terminated` (oder `Terminating`) widerspiegeln.

Variationen mit lokalen

Bei der Verwendung von Amazon GameLift Servers Local sollten Sie Folgendes beachten:

- Anders als der Amazon GameLift Servers-Web-Service verfolgt Local nicht den Zustand eines Servers und initiiert den `onProcessTerminate`-Callback nicht. Local beendet lediglich die Protokollierung für den Zustand des Spiel-Servers.
- Bei Aufrufen an das AWS SDK IDs werden Fleet nicht validiert und es kann sich um einen beliebigen Zeichenkettenwert handeln, der die Parameteranforderungen (`^fleet-\S+`) erfüllt.

- Mit Local IDs erstellte Spielsitzungen haben eine andere Struktur. Sie enthalten die Zeichenfolge `local`, wie hier dargestellt:

```
arn:aws:gamelift:local::gamesession/fleet-123/gsess-56961f8e-  
db9c-4173-97e7-270b82f0daa6
```

Debuggen von Amazon GameLift Servers-Flottenproblemen

Dieses Thema enthält Anleitungen zur Lösung von Problemen mit Ihren Amazon GameLift Servers verwalteten EC2 Flotten.

Probleme bei der Erstellung von Flotten

Wenn Sie eine verwaltete EC2 Flotte erstellen, initiiert der Amazon GameLift Servers Service einen Workflow, der die Flotte erstellt, EC2 Instances bereitstellt, auf denen Ihr Spieleserver-Build installiert ist, und Spieleserver-Prozesse auf jeder Instanz startet. Eine ausführliche Beschreibung finden Sie unter. Eine Flotte kann erst dann Spielsitzungen und Spieler hosten, wenn sie den Status Aktiv erreicht hat.

Sie können Probleme beheben, die verhindern, dass Flotten aktiv werden, indem Sie die Phase der Flottenerstellung identifizieren, in der das Problem aufgetreten ist, und die Ereignisse und Protokolle zur Flottenerstellung überprüfen. Wenn die Protokolle keine nützlichen Informationen enthalten, ist das Problem möglicherweise auf einen internen Servicefehler zurückzuführen. Versuchen Sie in diesem Fall erneut, die Flotte zu erstellen. Wenn das Problem weiterhin besteht, versuchen Sie, den Spiel-Build erneut hochzuladen, um eine mögliche Dateibeschädigung zu beheben). Sie können sich auch an den Amazon GameLift Servers-Support wenden oder im Forum eine Frage stellen.

Den Build herunterladen und validieren

Amazon GameLift Servers ruft während dieser Phase den hochgeladenen Spieleserver-Build ab, extrahiert die Dateien und führt alle Installationsskripte aus. Wenn die Flottenerstellung in diesen Phasen fehlschlägt, schauen Sie sich die Flottenereignisse und -protokolle an, um das Problem zu lokalisieren. Mögliche Gründe hierfür sind:

- Amazon GameLift Servers kann die komprimierte Build-Datei (Ereignis `FLEET_BINARY_DOWNLOAD_FAILED`) nicht abrufen. Stellen Sie sicher, dass auf den Speicherort des Builds zugegriffen werden kann, dass Sie eine Flotte erstellen, die dem Build entspricht AWS-Region , und dass Sie Amazon GameLift Servers über die richtigen Zugriffsberechtigungen verfügen.

- Amazon GameLift Servers kann die Build-Dateien nicht extrahieren (Ereignis `FLEET_CREATION_EXTRACTING_BUILD`).
- Ein Installationsskript in den Build-Dateien konnte nicht erfolgreich abgeschlossen werden (Ereignis `FLEET_CREATION_FAILED_INSTALLER`).

Ressourcen für die Flotte aufbauen

Probleme in dieser Phase betreffen in der Regel die Zuweisung und den Einsatz von Flottenressourcen. Mögliche Gründe hierfür sind:

- Der angeforderte Instance-Typ ist nicht verfügbar.
- Der angeforderte Flottentyp (Spot oder On-Demand) ist nicht verfügbar.

Gameserver-Prozesse aktivieren

In dieser Phase werden eine Reihe von Aufgaben durchgeführt und wichtige Elemente getestet, darunter die Funktionsfähigkeit des Spielservers, die Einstellungen für die Laufzeitkonfiguration und die Fähigkeit des Spielservers, mithilfe des Server-SDK eine Verbindung zum Amazon GameLift Servers Dienst herzustellen. Amazon GameLift Servers

Note

In dieser Phase können Sie remote auf eine Flotteninstanz zugreifen, um Probleme weiter zu untersuchen. Siehe [Stellen Sie eine Remoteverbindung zu Amazon GameLift Servers Flotteninstanzen her](#).

Zu den möglichen Problemen gehören:

- Serverprozesse werden nicht gestartet. Dies deutet auf ein Problem mit den Laufzeitkonfigurationseinstellungen der Flotte hin (Ereignisse `FLEET_VALIDATION_LAUNCH_PATH_NOT_FOUND` oder `FLEET_VALIDATION_EXECUTABLE_RUNTIME_FAILURE`. Stellen Sie sicher, dass Sie den Startpfad und die optionalen Startparameter korrekt eingestellt haben.
- Serverprozesse werden gestartet, aber die Flotte kann nicht aktiviert werden. Wenn Serverprozesse erfolgreich gestartet und ausgeführt werden, die Flotte jedoch nicht in den Status Aktiv wechselt, liegt das wahrscheinlich daran, dass der Serverprozess nicht mit dem Amazon GameLift Servers Dienst kommunizieren kann. Vergewissere dich, dass dein Spieleserver die richtigen Server-SDK-Aufrufe durchführt (siehe [Initialisieren Sie den Serverprozess](#)):

- Der Serverprozess kann nicht initialisiert werden (EreignisSERVER_PROCESS_SDK_INITIALIZATION_TIMEOUT). Der Serverprozess ruft `InitSdk()` nicht erfolgreich auf.
- Der Serverprozess benachrichtigt nicht, Amazon GameLift Servers wenn er bereit ist, eine Spielsitzung (ein EreignisSERVER_PROCESS_PROCESS_READY_TIMEOUT) zu veranstalten. Der Serverprozess wurde initialisiert, aber nicht `ProcessReady()` rechtzeitig aufgerufen.
- Eine VPC-Peering-Verbindungsanfrage ist fehlgeschlagen. Für Flotten, die mit einer VPC-Peering-Verbindung erstellt werden (siehe [So richten Sie VPC-Peering mit einer neuen Flotte ein](#)), erfolgt das VPC-Peering während dieser Activating (Aktivierung) Phasen. Wenn ein VPC-Peering aus irgendeinem Grund fehlschlägt, kann die neue Flotte nicht in den Status Active (Aktiv) wechseln. Sie können den Erfolg oder Misserfolg der Peering-Anfrage nachverfolgen, indem Sie aufrufen. [describe-vpc-peering-connections](#) Stellen Sie sicher, dass eine gültige VPC-Peering-Autorisierung vorhanden ist ([describe-vpc-peering-authorizations](#), da Autorisierungen nur für 24 Stunden gültig sind).

Probleme mit dem Serverprozess

Die Serverprozesse starten, werden jedoch nach kurzer Zeit wieder beendet oder laufen nicht richtig rund.

Ein solches Systemverhalten muss nicht zwangsläufig auf Probleme mit Ihrem Spiele-Build hinweisen, sondern es kann sich auch ergeben, wenn auf der Instance zu viele Serverprozesse gleichzeitig ausgeführt werden. Die optimale Anzahl an gleichzeitigen Prozessen hängt gleichermaßen von den Instance-Typ und den Ressourcenanforderungen Ihrer Spiel-Server ab. Verringern Sie die Anzahl der gleichzeitigen Prozesse, die in der Laufzeitkonfiguration der Flotte eingestellt wird, um zu prüfen, ob sich die Leistung verbessert. Sie können die Laufzeitkonfiguration einer Flotte entweder über die Amazon GameLift Servers Konsole ändern (bearbeiten Sie die Einstellungen für die Kapazitätszuweisung der Flotte) oder indem Sie den Befehl aufrufen. AWS CLI [update-runtime-configuration](#)

Probleme bei der Flottenlöschung

Die Flotte kann aufgrund der maximalen Anzahl von Instances nicht beendet werden.

Die Fehlermeldung zeigt an, dass die zu löschende Flotte noch aktive Instances hat, was nicht zulässig ist. Sie müssen zunächst eine Flotte auf null aktive Instances reduzieren. Dies geschieht,

indem Sie die gewünschte Anzahl der Instances der Flotte manuell auf "0" festlegen und dann darauf warten, dass die Skalierung greift. Stellen Sie sicher, dass Sie die auto Skalierung ausschalten, da dies manuellen Einstellungen entgegenwirkt.

VPC-Aktionen sind nicht autorisiert.

Dieses Problem gilt nur für Flotten, für die Sie speziell VPC-Peering-Verbindungen erstellt haben (siehe. [VPC-Peering einrichten für Amazon GameLift Servers](#)). Dieses Szenario tritt auf, weil das Löschen einer Flotte auch das Löschen der VPC der Flotte und aller VPC-Peering-Verbindungen umfasst. Sie müssen zuerst eine Autorisierung einholen, indem Sie die Service-API for Amazon GameLift Servers [CreateVpcPeeringAuthorization\(\)](#) aufrufen oder den AWS CLI-Befehl verwenden `create-vpc-peering-authorization`. Sobald Sie die Berechtigung haben, können Sie die Flotte löschen.

Stellen Sie eine Remoteverbindung zu Amazon GameLift Servers Flotteninstanzen her

Sie können eine Verbindung zu jeder Instance in Ihren aktiven Amazon GameLift Servers verwalteten EC2- oder Container-Flotten herstellen. Zu den häufigsten Gründen für den Fernzugriff auf eine Instance gehören:

- Beheben Sie Probleme mit der Integration Ihres Spieleservers.
- Optimiere deine Runtime-Konfiguration und andere flottenspezifische Einstellungen.
- Erhalte Gameserver-Aktivitäten in Echtzeit, z. B. Log-Tracking.
- Führen Sie Benchmarking-Tools anhand des tatsächlichen Spielerverkehrs aus.
- Untersuchen Sie spezifische Probleme mit einer Spielsitzung oder einem Serverprozess.

Wenn du eine Verbindung zu einer Instanz herstellst, solltest du die folgenden potenziellen Probleme berücksichtigen:

- Sie können eine Verbindung zu jeder Instance in einer aktiven Flotte herstellen. Im Allgemeinen können Sie keine Verbindung zu inaktiven Flotten herstellen, z. B. zu Flotten, die gerade aktiviert werden oder sich in einem Fehlerstatus befinden. (Diese Flotten sind möglicherweise für einen kurzen Zeitraum nur begrenzt verfügbar.) Hilfe bei Problemen mit der Flottenaktivierung finden Sie unter [Debuggen von Amazon GameLift Servers-Flottenproblemen](#).

- Die Verbindung zu einer aktiven Instance hat keinen Einfluss auf die Hosting-Aktivität der Instance. Die Instanz startet und stoppt weiterhin Serverprozesse auf der Grundlage der Laufzeitkonfiguration. Sie aktiviert und führt Spielsitzungen aus. Die Instanz wird möglicherweise als Reaktion auf ein Herunterskalierungsereignis oder ein anderes Ereignis heruntergefahren.
- Alle Änderungen, die Sie an Dateien oder Einstellungen auf der Instanz vornehmen, können sich auf die aktiven Spielsitzungen der Instanz und auf die verbundenen Spieler auswirken.

Fernzugriff über die Konsole

Mit Amazon EC2 Systems Manager (SSM) können Sie direkt von der Amazon GameLift Servers Konsole aus eine Verbindung zu Flotteninstanzen herstellen. Diese Methode bietet sicheren Zugriff, ohne dass eine zusätzliche Einrichtung oder Verwaltung der Anmeldeinformationen erforderlich ist.

1. Wählen Sie in der Amazon GameLift Servers Konsole im Navigationsbereich Managed EC2 oder Managed Containers und dann Fleets aus.
2. Wählen Sie die Flotten-ID aus, die die Instance enthält, auf die Sie zugreifen möchten.
3. Wählen Sie auf der Seite mit den Flottendetails den Tab Instances, um alle Compute-Instances für die Flotte anzuzeigen.
4. Wählen Sie die Instance aus, zu der Sie eine Verbindung herstellen möchten, und wählen Sie dann Connect. Daraufhin wird das Dialogfeld Mit Instanz Connect angezeigt, das Sie über die Details der Verbindung informiert und es Ihnen ermöglicht, das Skript anzuzeigen, das für die Verbindung mit Ihrer Instance verwendet wird. Bestätigen Sie, indem Sie erneut Verbinden wählen.
5. Wählen Sie im Verbindungsdialogfeld die Option Ausführen, um eine neue SSM-Sitzung zu erstellen. Das System authentifiziert Ihre Sitzung über den AWS Key Management Service (AWS KMS) und öffnet ein Terminal in Ihrem Browser.

Note

Konsolenbasierter Fernzugriff ist für Flotten verfügbar, auf denen das Server-SDK Version 5.x ausgeführt wird. Verwenden Sie für Flotten, auf denen frühere SDK-Versionen ausgeführt werden, die im folgenden Abschnitt beschriebene AWS CLI-Methode.

Fernzugriff mit der AWS CLI

In den folgenden Anweisungen wird beschrieben, wie Sie mithilfe der AWS Befehlszeilenschnittstelle (CLI) eine Remoteverbindung zu einer Instance herstellen. Sie können mithilfe des AWS SDK auch programmatische Aufrufe tätigen, wie in der [Service-API-Referenz für Amazon GameLift Servers](#) dokumentiert.

Sammeln Sie Instanzdaten

Um eine Verbindung zu einer Amazon GameLift Servers verwalteten EC2-Flotteninstanz herzustellen, benötigen Sie die folgenden Informationen:

- Die ID der Instance, zu der Sie eine Verbindung herstellen möchten. Sie können entweder die Instanz-ID oder den ARN verwenden.
- das Server-SDK für die Amazon GameLift Servers Version, die auf der Instanz verwendet wird. Das Server-SDK ist in den Spiel-Build integriert, der auf der Instanz ausgeführt wird.

In den folgenden Anweisungen wird beschrieben, wie Sie diese Aufgaben mithilfe der AWS CLI ausführen. Sie müssen die Flotten-ID für die Instance kennen, zu der Sie eine Verbindung herstellen möchten.

1. Rufen Sie den Computernamen ab. Ruft eine Liste aller aktiven Computer in der Flotte ab. Rufen Sie [list-compute](#) mit einer Flotten-ID oder einem ARN auf. Geben Sie für eine Flotte mit einem einzigen Standort nur die Flottenkennung an. Geben Sie für eine Flotte mit mehreren Standorten die Flottenkennung und einen Standort an. Gibt bei verwalteten EC2-Flotten eine Liste von Flotteninstanzen `list-compute` zurück, und die Eigenschaft `ComputeName` ist die Instance-ID. Suchen Sie den Computer, auf den Sie zugreifen möchten.

Anforderung

```
aws gamelift list-compute \  
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \  
  --location ""sa-east-1"
```

Antwort

```
{  
  "ComputeList": [  
    {
```

```

    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetArn": "arn:aws:gamelift:us-west-2::fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "ComputeName": "i-0abc12d3e45fa6b78",
    "IpAddress": "00.00.000.00",
    "DnsName":
    "b08444ki909kvqu6zpw3is24x5pyz4b6m05i3jbxvpk9craztu0lqrbbrbnbkks.uwp57060n1k6dn1nw49b78hg1
west-2.amazongamelift.com",
    "ComputeStatus": "Active",
    "Location": "sa-east-1",
    "CreationTime": "2023-07-09T22:51:45.931000-07:00",
    "OperatingSystem": "AMAZON_LINUX_2023",
    "Type": "c4.large"
  }
]
}

```

2. Suchen Sie die Server-SDK-Version. Um diese Informationen zu erhalten, müssen Sie nach dem Build suchen, das für die Flotte eingesetzt wird. Die Server-SDK-Version ist eine Build-Eigenschaft.
 - a. Rufen Sie [describe-fleet-attributes](#) mit einer Flotten-ID oder einem ARN an, um die Build-ID und den ARN der Flotte zu erhalten.
 - b. Rufen Sie [describe-build](#) mit der Build-ID oder dem ARN auf, um die Server-SDK-Version des Builds abzurufen.

Beispiel:

Anforderung

```

aws gamelift describe-fleet-attributes /
--fleet-ids "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"

```

Antwort

```

{
  "FleetAttributes": [
    {
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "ComputeType": "EC2",
      "BuildId": "build-3333cccc-44dd-55ee-66ff-00001111aa22",

```

```
    . . .  
  }  
]  
}
```

Anforderung

```
aws gamelift describe-build /  
  --build-id "build-3333cccc-44dd-55ee-66ff-00001111aa22"
```

Antwort

```
"Build": {  
  "BuildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",  
  "Name": "My_Game_Server_Build_One",  
  "OperatingSystem": "AMAZON_LINUX_2023",  
  "ServerSdkVersion": "5.1.1",  
  . . .  
}
```

Connect zu einer Instanz herstellen (Server-SDK 5)

Wenn auf der Instance, zu der Sie eine Verbindung herstellen möchten, ein Game-Build mit Server-SDK Version 5.x ausgeführt wird, stellen Sie mithilfe von Amazon EC2 Systems Manager (SSM) eine Verbindung zu der Instance her. Sie können auf Remote-Instances zugreifen, die auf Windows oder Linux ausgeführt werden.

Bevor Sie beginnen:

Schließen Sie die SSM-Setup-Schritte ab und installieren Sie das SSM-Plugin auf Ihrem lokalen Computer. Weitere Informationen finden Sie unter [SSM einrichten](#) und [das Session Manager-Plug-In für die AWS CLI installieren](#) im Amazon EC2 Systems Manager Manager-Benutzerhandbuch.

1. Fordern Sie Anmeldeinformationen für die Instance an. Rufen Sie [get-compute-access](#) mit der Flotten-ID und dem Rechenamen für die Instance an, zu der Sie eine Verbindung herstellen

möchten. Amazon GameLift Servers gibt einen Satz temporärer Anmeldeinformationen für den Zugriff auf die Instance zurück. Beispiel:

Anforderung

```
aws gamelift get-compute-access \  
--compute-name i-11111111a222b333c \  
--fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
--region us-west-2
```

Antwort

```
{  
  "ComputeName": " i-11111111a222b333c ",  
  "Credentials": {  
    "AccessKeyId": " ASIAIOSFODNN7EXAMPLE ",  
    "SecretAccessKey": " wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY ",  
    "SessionToken": " AQoDYXdzEJr...<remainder of session token>"  
  },  
  "FleetArn": " arn:aws:gamelift:us-west-2::fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa ",  
  "FleetId": " fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa "  
}
```

2. Exportieren Sie die Zugangsdaten (optional). Sie können die Anmeldeinformationen in Umgebungsvariablen exportieren und sie verwenden, um die AWS CLI für den Standardbenutzer zu konfigurieren. Weitere Informationen finden Sie unter [Umgebungsvariablen zur Konfiguration der AWS CLI](#) im AWS Command Line Interface Benutzerhandbuch.

```
export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE  
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY  
export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
```

3. Connect zur Flotteninstanz her. Starten Sie eine SSM-Sitzung mit der Instance, zu der Sie eine Verbindung herstellen möchten. Geben Sie die AWS Region oder den Standort der Instanz an. Weitere Informationen, einschließlich der Einrichtung von SSM und des SSM-Plug-ins, finden Sie unter [Starten einer Sitzung \(AWS CLI\)](#) im Amazon EC2 Systems Manager Benutzerhandbuch.

Für die Anfrage zum Sitzungsstart werden automatisch die Anmeldeinformationen verwendet, die Sie in Schritt 1 erworben haben.


```
aws ssm start-session \  
--target i-11111111a222b333c \  
--region us-west-2 \
```

Note

Wenn Sie die Fehlermeldung „Zugriff verweigert“ erhalten, haben Sie möglicherweise eine `AWS_PROFILE` Umgebungsvariable auf ein AWS Profil gesetzt, was dazu führt, dass AWS CLI die falschen Anmeldeinformationen für den Remotezugriff verwendet. Um das Problem zu beheben, deaktivieren Sie vorübergehend Ihre `AWS_PROFILE` Umgebungsvariable. Alternativ können Sie ein benutzerdefiniertes AWS Profil für Ihre RAS-Anmeldeinformationen erstellen und den `--profile` Befehlszeilenparameter zu Ihrer `start-session` Anfrage hinzufügen.

Stellen Sie eine Connect zu einer Instanz her (Server-SDK 4.x oder früher)

Wenn auf der Instance, zu der Sie eine Verbindung herstellen möchten, ein Spiel mit Server-SDK Version 4 oder früher ausgeführt wird, gehen Sie wie folgt vor. Sie können eine Verbindung zu Instances herstellen, auf denen entweder Windows oder Linux ausgeführt wird. Stellen Sie mithilfe eines RDP-Clients (Remote Desktop Protocol) eine Connect zu einer Windows-Instanz her. Stellen Sie mithilfe eines SSH-Clients eine Connect zu einer Linux-Instance her.

1. Fordern Sie Anmeldeinformationen für die Instance an. Wenn Sie eine Instanz-ID haben, verwenden Sie den Befehl, [get-instance-access](#) um Zugangsdaten anzufordern. Wenn dies erfolgreich ist, gibt Amazon GameLift Servers das Betriebssystem, die IP-Adresse und einen Satz von Anmeldeinformationen (Benutzername und geheimer Schlüssel) für die Instance zurück. Das Format der Anmeldeinformationen ist vom Betriebssystem der Instance abhängig. Verwenden Sie die folgenden Anweisungen, um Anmeldeinformationen für RDP oder SSH abzurufen.
 - Für Windows-Instanzen — Um eine Verbindung zu einer Windows-Instanz herzustellen, benötigt RDP einen Benutzernamen und ein Passwort. Die `get-instance-access`-Anforderung gibt diese Werte als einfache Zeichenfolgen zurück. Daher können Sie die zurückgegebenen Werte unverändert verwenden. Beispiel für Anmeldeinformationen:

```
"Credentials": {
```

```
"Secret": "aA1bBB2cCCd3EEE",
"UserName": "gl-user-remote"
}
```

- Für Linux-Instances — Um eine Verbindung zu einer Linux-Instance herzustellen, benötigt SSH einen Benutzernamen und einen privaten Schlüssel. Amazon GameLift Servers gibt private RSA-Schlüssel aus und gibt sie als einzelne Zeichenfolge zurück, wobei das Zeilenumbruchzeichen (\n) auf Zeilenumbrüche hinweist. Gehen Sie wie folgt vor, um den privaten Schlüssel nutzbar zu machen: (1) konvertieren Sie die Zeichenfolge in eine .pem Datei und (2) legen Sie die Berechtigungen für die neue Datei fest. Beispiel für zurückgegebene Anmeldeinformationen:

```
"Credentials": {
  "Secret": "-----BEGIN RSA PRIVATE KEY-----
nEXAMPLEKEYKCAQEay7WZhaDsra1W3mRlQtvhwyORRX8gnxgDaFRt/gx42kWXsT4rXE/b5CpSgie/
\nvBoU7jLxx92pNHoFnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7i05dSrvC7dQkW2duV5QuUdEQW
\nZ/aNxMniGQE6XAgfwlnXVBwrerrQo+ZWQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449i1x9X1F
\nG50TCFe0zfl8dqqCP6GzbPaIjiU19xX/az0R9V+tpU0zEL+wmXnZt3/nHPQ5xvD20JH67km6SuPW
\noPzev/D8V+x4+bHthfSjR9Y7DvQfjFBVwHXigBdtZcU2/wei8D/HYwIDAQABaoIBAGZ1kaEvnrrqu
\n/uler7vgIn5m71N5LKw4hJLAIW6tUT/fzvtcHK0SkbQCQXuriHmQ2MqyJX/0kn2NfjLV/
ufGxbL1\nmb5qwmGUnEpJaZD6QSSs3kICLwUUYUiGfc0uiSbmJoap/
GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2\nbahyWyJNfjLe4M86yd2YK3V2CmK+X/
B0sShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/
tJWSD9\n81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9Jka30zdXzMqexXVJ1TLZVEH0E7bh1Y9d801ozR
\noQs/FiZNAx2iijCWyv01pjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfq1
+1Ip1\nYkriL0DbLXlvRAH+yHPRit2hH0jtUNZh4Axv+cpg09qbUI3+43eEy24B7G/Uh
+GTfbjsXs0xQx/x\np9otyVwc7hsQ5TA5PZb
+mvkJ50BEKzet9XcKw0NBVELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws
\nayav0elc5zkxjF9nfHfJRry21R1trw2Vdpn+9g481URrpzWV0Eihvm+XTtmaZ1Sp//1kq75XDwnU
\nWA8gkn603QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Unl2ajLivWUt5pbBrKbUC
\nngYBjb0+0Zk0sCcpZ29sbzjYjpIddErySIyRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
\nnoMo7yykABY70zd5wQewBQ4AdSlWSX4nGDtsiFxiW5sKuAAe0CbTosy1s8w8fxoJ5Tz1sdoxNeGs
\nArq6Wv/G16zQuAE9zK9vwwKBgF+09VI/1wJBirsDGz9whVwFFPrTkJNvJZzYt69qezx1sjgFKshy
\nWBhd4xHZtmCqpBP1AymEjr/T01bxyARmXMnIOWIANNXMGB4KGSyl1mzSVAoQ+fqR+cJ3d0dyP11j
\nnjjb0Ed/NY8fr1NDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0i0egLda
\nNWUUh38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvtW+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
\nVRkAKKKYeGjKpUfVTrW0YFjXkfcR/V+QFL50ndHAKJXjW7a4ejJLncTzmZSpYzwApc=\n-----END
RSA PRIVATE KEY-----",
  "UserName": "gl-user-remote"
}
```

Wenn Sie die AWS CLI verwenden, können Sie automatisch eine .pem Datei generieren, indem Sie die Parameter `--query` und `--output` in Ihre `get-instance-access` Anfrage aufnehmen.

Um für die .pem-Datei Berechtigungen festzulegen, führen Sie den folgenden Befehl aus:

```
$ chmod 400 MyPrivateKey.pem
```

2. Öffnen Sie einen Port für die Remote-Verbindung. Sie können über jeden in der Amazon GameLift Servers Flottenkonfiguration autorisierten Port auf Instances in Flotten zugreifen. Sie können die Port-Einstellungen einer Flotte mit dem Befehl [describe-fleet-port-settings](#) anzeigen.

Als bewährte Methode wird empfohlen, Ports für den Remote-Zugriff nur zu öffnen, wenn Sie diese benötigen, und sie zu schließen, wenn Sie diese nicht mehr benötigen. Sie können die Porteinstellungen nicht aktualisieren, nachdem Sie eine Flotte erstellt haben, sondern bevor sie aktiv ist. Wenn Sie nicht weiterkommen, erstellen Sie die Flotte mit geöffneten Porteinstellungen neu.

Verwenden Sie den Befehl [update-fleet-port-settings](#), um eine Port-Einstellung für die Remote-Verbindung hinzuzufügen (z. B. 22 für SSH oder 3389 für RDP). Geben Sie als Wert für den IP-Adressbereich die IP-Adressen für die Geräte an, die Sie verwenden möchten, um eine Verbindung herzustellen (in das CIDR-Format konvertiert). Beispiel:

```
$ AWS gamelift update-fleet-port-settings
--fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
--inbound-permission-authorizations
"FromPort=22,ToPort=22,IpRange=54.186.139.221/32,Protocol=TCP"
```

Das folgende Beispiel öffnet Port 3389 auf einer Windows-Flotte

```
$ AWS gamelift update-fleet-port-settings
--fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
--inbound-permission-authorizations
"FromPort=3389,ToPort=3389,IpRange=54.186.139.221/32,Protocol=TCP"
```

3. Öffnen Sie einen Remote-Verbindungs-Client. Verwenden Sie Remote Desktop für Windows-Instances oder SSH für Linux-Instances. Stellen Sie die Verbindung mit der Instance unter Verwendung von IP-Adresse, Port-Einstellung und Anmeldeinformationen her.

SSH-Beispiel:

```
ssh -i MyPrivateKey.pem gl-user-remote@192.0.2.0
```

Dateien auf Remote-Instances anzeigen

Unabhängig davon, ob Sie über die Konsole oder die AWS CLI eine Remoteverbindung zu einer Instance hergestellt haben, haben Sie vollen Benutzer- und Administratorzugriff darauf. Das bedeutet, dass Sie auch beim Hosten von Spielen Fehler oder Ausfälle verursachen können. Wenn die Instanz Spiele mit aktiven Spielern hostet, besteht die Gefahr, dass Spielsitzungen abstürzen und Spieler fallen gelassen werden oder dass das Herunterfahren von Spielen unterbrochen wird, was zu Fehlern in den gespeicherten Spieldaten und Protokollen führen kann.

Suchen Sie auf einer Hosting-Instanz nach diesen Ressourcen:

- Build-Dateien von Spielen. Diese Dateien sind der Spiel-Build, in den Sie hochgeladen haben Amazon GameLift Servers. Sie enthalten eine oder mehrere ausführbare Dateien, Ressourcen und Abhängigkeiten des Spielservers. Die Build-Dateien des Spiels befinden sich in einem Stammverzeichnis mit dem Namen: game
 - Bei Windows: c:\game
 - Bei Linux: /local/game
- Protokolldateien von Spielen. Suchen Sie die Protokolldateien, die Ihr Spieleserver generiert, im game Stammverzeichnis in dem von Ihnen angegebenen Verzeichnispfad.
- Hosting-Ressourcen von Amazon GameLift Servers. Das Stammverzeichnis Whitewater enthält Dateien, die vom Amazon GameLift Servers Dienst zur Verwaltung der Spielhosting-Aktivitäten verwendet werden. Verändern Sie diese Dateien aus keinem Grund.
- Laufzeitkonfiguration. Greifen Sie nicht auf die Laufzeitkonfiguration für einzelne Instanzen zu. Um Änderungen an einer Laufzeitkonfigurationseigenschaft vorzunehmen, aktualisieren Sie die Laufzeitkonfiguration der Flotte (siehe den AWS SDK-Vorgang [UpdateRuntimeConfiguration](#) oder den AWS CLI [update-runtime-configuration](#)).
- Flottendaten. Eine JSON-Datei enthält Informationen über die Flotte, zu der die Instance gehört, zur Verwendung durch Serverprozesse, die auf der Instance ausgeführt werden. Die JSON-Datei befindet sich am folgenden Speicherort:
 - Bei Windows: C:\GameMetadata\gamelift-metadata.json

- Bei Linux: `/local/gamemetadata/gamelift-metadata.json`
- TLS-Zertifikate. Wenn sich die Instanz auf einer Flotte befindet, für die die Generierung von TLS-Zertifikaten aktiviert ist, suchen Sie am folgenden Speicherort nach Zertifikatsdateien, einschließlich des Zertifikats, der Zertifikatskette, des privaten Schlüssels und des Stammzertifikats:
 - Bei Windows: `c:\\GameMetadata\\Certificates`
 - Bei Linux: `/local/gamemetadata/certificates/`

Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers

Die Hosting-Kapazität, gemessen in Instanzen, gibt die Anzahl der Spielsitzungen an, die gleichzeitig veranstaltet Amazon GameLift Servers werden können, und die Anzahl der gleichzeitigen Spieler, die diese Spielsitzungen aufnehmen können. Eine der schwierigsten Aufgaben beim Hosten von Spielen ist die Skalierung der Kapazität, um der Nachfrage der Spieler gerecht zu werden, ohne Geld für Ressourcen zu verschwenden, die Sie nicht benötigen.

Die Kapazität wird auf Ebene des Flottenstandorts angepasst. Alle Flotten haben mindestens einen Standort: die AWS Heimatregion der Flotte. Bei der Anzeige oder Skalierung der Kapazität werden die Informationen nach Standort aufgelistet, einschließlich der Heimatregion der Flotte und aller zusätzlichen abgelegenen Standorte.

Sie können die Anzahl der zu verwaltenden Instances manuell festlegen oder die Auto Scaling einrichten, um die Kapazität dynamisch anzupassen, wenn sich die Nachfrage der Spieler ändert. Wir empfehlen, dass Sie zunächst die Option für zielbasiertes Auto Scaling aktivieren. Das Ziel des zielgerichteten Auto Scaling besteht darin, genügend Hosting-Ressourcen für aktuelle Spieler und darüber hinaus zusätzliche Ressourcen bereitzustellen, um unerwartete Anstiege der Spielernachfrage zu bewältigen. Für die meisten Spiele bietet das zielgerichtete Auto Scaling eine hocheffektive Skalierungslösung.

Die meisten Aktivitäten zur Flottenskalierung können Sie über die Amazon GameLift Servers Konsole ausführen. Sie können auch ein AWS SDK oder das AWS Command Line Interface (AWS CLI) mit der [Service-API für](#) verwenden Amazon GameLift Servers.

Themen

- [Um die Flottenkapazität in der Konsole zu verwalten](#)
- [Amazon GameLift ServersKapazitätsgrenzen festlegen](#)
- [Manuelles Einstellen der Kapazität für eine Amazon GameLift Servers Flotte](#)
- [Automatische Skalierung der Flottenkapazität mit Amazon GameLift Servers](#)
- [Verwaltung von Scaling und Amazon GameLift Servers Fleet To/From Zero](#)
- [Amazon GameLift ServersContainerflotten skalieren](#)

Um die Flottenkapazität in der Konsole zu verwalten

1. Öffnen Sie die [Amazon GameLift Servers-Konsole](#).
2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Flottendetailseite zu öffnen.
4. Wählen Sie die Registerkarte Skalierung. Auf dieser Registerkarte können Sie:
 - Historische Skalierungskennzahlen für die gesamte Flotte anzeigen.
 - Zeigen Sie die Kapazitätseinstellungen für jeden Flottenstandort an und aktualisieren Sie sie, einschließlich Skalierungsgrenzen und aktueller Kapazitätseinstellungen.
 - Aktualisieren Sie zielbasiertes Auto Scaling, lassen Sie sich regelbasierte Auto-Scaling-Richtlinien anzeigen, die für die gesamte Flotte gelten, und unterbrechen Sie die Auto-Scaling-Aktivität für jeden Standort.

Amazon GameLift ServersKapazitätsgrenzen festlegen

Beachten Sie bei der manuellen oder auto Skalierung der Hosting-Kapazität für einen Amazon GameLift Servers Flottenstandort die Skalierungsgrenzen des Standorts. Für alle Flottenstandorte gibt es eine Mindest- und Höchstgrenze, die den zulässigen Bereich für die Kapazität des Standorts definieren. Standardmäßig haben Grenzwerte für Flottenstandorte ein Minimum von 0 Instances und ein Maximum von 1 Instance. Bevor Sie einen Flottenstandort skalieren können, müssen Sie die Grenzwerte anpassen.

Wenn Sie Auto Scaling verwenden, ermöglicht das Höchstlimit die Skalierung eines Flottenstandorts, Amazon GameLift Servers um der Nachfrage der Spieler gerecht zu werden, verhindert jedoch, dass die Hosting-Kosten, wie z. B. bei einem DDOS-Angriff, außer Kontrolle geraten. Richten Sie einen [CloudWatch Amazon-Alarm](#) ein, der Sie benachrichtigt, wenn sich die Kapazität der Höchstgrenze nähert, sodass Sie die Situation bewerten und bei Bedarf manuell anpassen können. (Sie können auch [einen Abrechnungsalarm einrichten](#), um die AWS Kosten zu überwachen.) Das Mindestlimit ist nützlich, um die Verfügbarkeit des Hostings aufrechtzuerhalten, auch wenn die Nachfrage der Spieler gering ist.

Sie können Kapazitätsgrenzen für die Standorte einer Flotte in der [Amazon GameLift ServersKonsole](#) oder mithilfe von AWS Command Line Interface (AWS CLI) festlegen.

So legen Sie die Grenzwerte für die Flottenkapazität fest

Console

1. Öffnen Sie die [Amazon GameLift Servers-Konsole](#).
2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Flottendetailseite zu öffnen.
4. Wählen Sie auf der Registerkarte Skalierung unter Skalierung der Kapazität einen Flottenstandort aus und klicken Sie dann auf Bearbeiten.
5. Legen Sie im Dialogfeld Skalierungskapazität bearbeiten die Anzahl der Instanzen für Mindestgröße, Gewünschte Instances und Maximale Größe fest.
6. Wählen Sie Bestätigen aus.

AWS CLI

1. Überprüfen Sie die aktuellen Kapazitätseinstellungen. Verwenden Sie in einem Befehlszeilenfenster den [describe-fleet-location-capacity](#)Befehl mit der Flotten-ID und dem Standort, für den Sie die Kapazität ändern möchten. Dieser Befehl gibt ein [FleetCapacity](#)Objekt zurück, das die aktuellen Kapazitätseinstellungen des Standorts enthält. Ermitteln Sie, ob die neuen Instance-Limits der aktuellen Einstellung für die gewünschten Instanzen entsprechen.

```
aws gamelift describe-fleet-location-capacity \
  --fleet-id <fleet identifier> \
  --location <location name>
```

2. Aktualisieren der Limiteinstellungen. Verwenden Sie in einem Befehlszeilenfenster den [update-fleet-capacity](#)Befehl mit den folgenden Parametern. Sie können die Instance-Limits und die Anzahl der gewünschten Instances mit demselben Befehl anpassen.

```
--fleet-id <fleet identifier>
--location <location name>
--max-size <maximum capacity for scaling>
--min-size <minimum capacity for scaling>
--desired-instances <fleet capacity goal>
```


Beispiel:

```
aws gamelift update-fleet-capacity \  
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --location us-west-2 \  
  --max-size 10 \  
  --min-size 1 \  
  --desired-instances 10
```

Wenn Ihre Anfrage erfolgreich ist, wird die Flotten-ID Amazon GameLift Servers zurückgegeben. Wenn der neue min-size Wert max-size oder mit der aktuellen desired-instances Einstellung in Konflikt steht, wird ein Fehler Amazon GameLift Servers zurückgegeben.

Manuelles Einstellen der Kapazität für eine Amazon GameLift Servers Flotte

Wenn Sie eine neue Flotte erstellen, ordnet Amazon GameLift Servers die gewünschten Instances automatisch einer Instance an jedem Flottenstandort zu. Amazon GameLift Servers stellt dann an jedem Standort eine neue Instance bereit. Um die Flottenkapazität zu ändern, können Sie eine zielbasierte Auto-Scaling-Richtlinie hinzufügen oder die Anzahl der Instanzen, die Sie für einen Standort benötigen, manuell festlegen.

Die manuelle Einstellung der Kapazität einer Flotte kann nützlich sein, wenn Sie keine auto Skalierung benötigen oder wenn Sie die Kapazität auf einem bestimmten Niveau halten müssen. Das manuelle Einstellen der Kapazität funktioniert nur, wenn Sie keine zielbasierte Auto Scaling-Richtlinie verwenden. Wenn Sie über eine zielbasierte Auto Scaling-Richtlinie verfügen, setzt diese sofort die gewünschte Kapazität auf der Grundlage ihrer eigenen Skalierungsregeln zurück.

Sie können die Kapazität manuell in der Amazon GameLift Servers Konsole oder mithilfe von AWS Command Line Interface (AWS CLI) festlegen. Der Status der Flotte muss aktiv sein.

Aussetzen des Auto Scaling

Sie können alle Auto-Scaling-Aktivitäten für jeden Flottenstandort aussetzen. Wenn das Auto Scaling ausgesetzt ist, bleibt die gewünschte Anzahl von Instances am Flottenstandort gleich, sofern sie nicht manuell geändert wird. Wenn Sie das Auto Scaling für einen Standort aussetzen, wirkt sich dies

auf die aktuellen Richtlinien der Flotte und alle Richtlinien aus, die Sie möglicherweise in Zukunft definieren.

So legen Sie die Flottenkapazität manuell fest

Console

1. Öffnen Sie die [Amazon GameLift Servers-Konsole](#).
2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Flottendetailseite zu öffnen.
4. Wählen Sie auf der Registerkarte Skalierung unter Unterbrochene Auto Scaling-Standorte jeden Standort aus, für den Sie Auto Scaling aussetzen möchten, und wählen Sie dann Suspend aus.
5. Wählen Sie unter Skalierungskapazität einen Standort aus, den Sie manuell festlegen möchten, und klicken Sie dann auf Bearbeiten.
6. Geben Sie im Dialogfeld Skalierungskapazität bearbeiten Ihren bevorzugten Wert für Gewünschte Instanzen ein und wählen Sie dann Bestätigen aus. Dies gibt die Anzahl Amazon GameLift Servers der Instances an, die im aktiven Zustand beibehalten werden sollen und bereit sind, Spielsitzungen abzuhalten.

Amazon GameLift Servers reagiert auf die Änderungen, indem zusätzliche Instanzen bereitgestellt oder nicht benötigte Instanzen heruntergefahren werden. Nach Amazon GameLift Servers Abschluss dieses Vorgangs ändert sich die Anzahl der aktiven Instanzen am Standort entsprechend dem aktualisierten Wert für die gewünschten Instanzen. Dieser Vorgang kann einige Zeit dauern.

AWS CLI

1. Überprüfen Sie die aktuellen Kapazitätseinstellungen. Verwenden Sie in einem Befehlszeilenfenster den [describe-fleet-location-capacity](#) Befehl mit der Flotten-ID und dem Standort, für den Sie die Kapazität ändern möchten. Dieser Befehl gibt ein [FleetCapacity](#) Objekt zurück, das die aktuellen Kapazitätseinstellungen des Standorts enthält. Stellen Sie fest, ob die Instanzlimits der neuen Einstellung für die gewünschten Instanzen entsprechen.

```
aws gamelift describe-fleet-location-capacity \
```

```
--fleet-id <fleet identifier> \  
--location <location name>
```

2. Aktualisierung der gewünschten Kapazität. Verwenden Sie den [update-fleet-capacity](#) Befehl mit der Flotten-ID, dem Standort und einem neuen Wert für die gewünschten Instances. Wenn dieser Wert außerhalb des aktuellen Grenzbereichs liegt, können Sie die Grenzwerte mit demselben Befehl anpassen.

```
--fleet-id <fleet identifier>  
--location <location name>  
--desired-instances <fleet capacity as an integer>  
--max-size <maximum capacity> [Optional]  
--min-size <minimum capacity> [Optional]
```

Beispiel:

```
aws gamelift update-fleet-capacity \  
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --location us-west-2 \  
  --desired-instances 5 \  
  --max-size 10 \  
  --min-size 1
```

Wenn Ihre Anfrage erfolgreich ist, wird die Flotten-ID Amazon GameLift Servers zurückgegeben. Wenn die neue Einstellung für die gewünschten Instanzen außerhalb der Mindest- und Höchstgrenzen liegt, wird ein Fehler Amazon GameLift Servers zurückgegeben.

Automatische Skalierung der Flottenkapazität mit Amazon GameLift Servers

Verwende Auto Scaling in Amazon GameLift Servers, um deine Flottenkapazität als Reaktion auf die Aktivität des Spielerservers dynamisch zu skalieren. Wenn Spieler ankommen und Spielsitzungen starten, kann Auto Scaling weitere Instanzen hinzufügen. Wenn die Nachfrage der Spieler nachlässt, kann Auto Scaling nicht benötigte Instanzen beenden. Auto Scaling ist eine effektive Methode, um Ihre Hosting-Ressourcen und -Kosten zu minimieren und gleichzeitig ein reibungsloses, schnelles Spielerlebnis zu bieten.

Um Auto Scaling zu verwenden, erstellen Sie Skalierungsrichtlinien, die angeben, Amazon GameLift Servers wann die Skalierung nach oben oder unten erfolgen soll. Es gibt zwei Arten von Skalierungsrichtlinien: zielbasierte und regelbasierte. Der zielbasierte Ansatz — Target Tracking — ist eine Komplettlösung. Wir empfehlen ihn als einfachste und effektivste Option. Bei regelbasierten Skalierungsrichtlinien müssen Sie jeden Aspekt des Entscheidungsprozesses für die auto Skalierung definieren, was für die Lösung bestimmter Probleme nützlich ist. Diese Lösung eignet sich am besten als Ergänzung zur zielbasierten auto Skalierung.

Sie können zielbasiertes Auto Scaling mit der Amazon GameLift Servers Konsole, dem AWS Command Line Interface (AWS CLI) oder einem AWS SDK verwalten. Sie können regelbasiertes Auto Scaling nur mit dem AWS CLI oder einem AWS SDK verwalten, obwohl Sie regelbasierte Skalierungsrichtlinien in der Konsole anzeigen können.

Themen

- [Zielbasiertes Auto Scaling](#)
- [Auto Scaling mit regelbasierten Richtlinien](#)

Zielbasiertes Auto Scaling

Zielbasierte auto Skalierung zur Amazon GameLift Servers Anpassung des Kapazitätsniveaus auf der Grundlage der Flottenmetrik. `PercentAvailableGameSessions` Diese Metrik stellt den verfügbaren Puffer der Flotte für plötzliche Anstiege der Spielernachfrage dar.

Der primäre Grund für die Einrichtung eines Kapazitätspuffers ist die Wartezeit der Spieler. Wenn die Slots für Spielsitzungen bereit sind und auf sie warten, dauert es Sekunden, bis neue Spieler in die Spielsitzungen aufgenommen werden. Wenn keine Ressourcen verfügbar sind, müssen die Spieler darauf warten, dass bestehende Spielsitzungen beendet werden, oder bis neue Ressourcen verfügbar sind. Das Starten neuer Instanzen und Serverprozesse kann Minuten dauern.

Wenn Sie zielbasiertes Auto Scaling einrichten, geben Sie die Größe des Puffers an, den die Flotte beibehalten soll. Da `PercentAvailableGameSessions` der Prozentsatz der verfügbaren Ressourcen gemessen wird, ist die tatsächliche Puffergröße ein Prozentsatz der gesamten Flottenkapazität. Amazon GameLift Servers fügt Instanzen hinzu oder entfernt sie, um die Zielpuffergröße beizubehalten. Mit einem großen Puffer minimieren Sie die Wartezeit, zahlen aber auch für zusätzliche Ressourcen, die Sie möglicherweise nicht nutzen. Wenn Ihre Spieler toleranter gegenüber Wartezeiten sind, können Sie die Kosten senken, indem Sie einen kleinen Puffer einrichten.

So legen Sie zielbasiertes Auto Scaling fest

Console

1. Öffnen Sie die [Amazon GameLift Servers-Konsole](#).
2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Flottendetailseite zu öffnen.
4. Wählen Sie die Registerkarte Skalierung. Diese Registerkarte zeigt die historischen Skalierungsmetriken der Flotte an und enthält Steuerelemente zum Anpassen der aktuellen Skalierungseinstellungen.
5. Vergewissern Sie sich unter Skalierung der Kapazität, dass die Grenzwerte Minimale Größe und Max. Größe für die Flotte angemessen sind. Wenn Auto Scaling aktiviert ist, passt sich die Kapazität zwischen diesen beiden Grenzwerten an.
6. Wählen Sie unter Zielbasierte Auto Scaling-Richtlinie die Option Bearbeiten aus.
7. Geben Sie im Dialogfeld Zielbasierte Auto Scaling-Richtlinie bearbeiten für Prozent verfügbarer Spielsitzungen den Prozentsatz ein, den Sie beibehalten möchten, und wählen Sie dann Bestätigen aus. Nachdem Sie die Einstellungen bestätigt haben, Amazon GameLift Servers fügt eine neue zielbasierte Richtlinie unter Zielbasierte Auto Scaling-Richtlinie hinzu.

AWS CLI

1. Festlegen der Grenzwerte für Flottenkapazität. Legen Sie die Grenzwerte mithilfe des [update-fleet-capacity](#)Befehls fest. Weitere Informationen finden Sie unter [Amazon GameLift ServersKapazitätsgrenzen festlegen](#).
2. Eine neue Richtlinie erstellen. Öffnen Sie ein Befehlszeilenfenster und verwenden Sie den [put-scaling-policy](#)Befehl mit den Parametereinstellungen Ihrer Richtlinie. Um eine bestehende Richtlinie zu aktualisieren, geben Sie den Namen der Richtlinie an und geben eine vollständige Version der aktualisierten Richtlinie an.

```
--fleet-id <unique fleet identifier>
--name "<unique policy name>"
--policy-type <target- or rule-based policy>
--metric-name <name of metric>
--target-configuration <buffer size>
```

Beispiel:

```
aws gamelift put-scaling-policy \  
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \  
  --name "My_Target_Policy_1" \  
  --policy-type "TargetBased" \  
  --metric-name "PercentAvailableGameSessions" \  
  --target-configuration "TargetValue=5"
```

Auto Scaling mit regelbasierten Richtlinien

Regelbasierte Skalierungsrichtlinien Amazon GameLift Servers bieten eine detaillierte Kontrolle bei der auto Skalierung der Kapazität einer Flotte als Reaktion auf Spieleraktivitäten. Für jede Richtlinie können Sie die Skalierung mit einer von mehreren Flottenkennzahlen verknüpfen, einen Triggerpunkt identifizieren und das darauf reagierende Scale-up- oder Scale-Down-Ereignis individuell anpassen. Regelbasierte Richtlinien sind nützlich, um die [zielbasierte](#) Skalierung zu ergänzen, um besonderen Umständen gerecht zu werden.

Eine regelbasierte Richtlinie besagt Folgendes: „Wenn eine Flottenkennzahl für einen bestimmten Zeitraum einen Schwellenwert erreicht oder überschreitet, muss die Kapazität der Flotte um einen bestimmten Betrag geändert werden.“ Dieses Thema beschreibt die Syntax zur Erstellung einer Richtlinienanweisung und bietet Hilfe bei der Erstellung und Verwaltung Ihrer regelbasierten Richtlinien.

Regelbasierte Richtlinien verwalten

[Erstellen, aktualisieren oder löschen Sie regelbasierte Richtlinien mithilfe eines AWS SDK oder des AWS Command Line Interface \(AWS CLI\) mit der Service-API für Amazon GameLift Servers](#) Sie können alle aktiven Richtlinien in der Amazon GameLift Servers-Konsole anzeigen.

Verwenden Sie den Befehl, um alle Skalierungsrichtlinien für eine Flotte vorübergehend zu beenden. AWS CLI [stop-fleet-actions](#)

So erstellen oder aktualisieren Sie eine regelbasierte Skalierungsrichtlinie (AWS CLI):

1. Festlegen der Grenzwerte für Flottenkapazität. Legen Sie mit dem Befehl einen oder beide Grenzwerte fest. [update-fleet-capacity](#) Weitere Informationen finden Sie unter [Amazon GameLift Servers Kapazitätsgrenzen festlegen](#).

2. Eine neue Richtlinie erstellen. Öffnen Sie ein Befehlszeilenfenster und verwenden Sie den [put-scaling-policy](#) Befehl mit den Parametereinstellungen Ihrer Richtlinie. Um eine bestehende Richtlinie zu aktualisieren, geben Sie den Namen der Richtlinie an und geben eine vollständige Version der aktualisierten Richtlinie an.

```
--fleet-id <unique fleet identifier>
--name "<unique policy name>"
--policy-type <target- or rule-based policy>
--metric-name <name of metric>
--comparison-operator <comparison operator>
--threshold <threshold integer value>
--evaluation-periods <number of minutes>
--scaling-adjustment-type <adjustment type>
--scaling-adjustment <adjustment amount>
```

Beispiel:

```
aws gamelift put-scaling-policy \
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --name "Scale up when AGS<50" \
  --policy-type RuleBased \
  --metric-name AvailableGameSessions \
  --comparison-operator LessThanThreshold \
  --threshold 50 \
  --evaluation-periods 10 \
  --scaling-adjustment-type ChangeInCapacity \
  --scaling-adjustment 1
```

Um eine regelbasierte Skalierungsrichtlinie zu löschen, verwenden Sie: AWS CLI

- Öffnen Sie ein Befehlszeilenfenster und verwenden Sie den [delete-scaling-policy](#) Befehl mit der Flotten-ID und dem Richtliniennamen.

Beispiel:

```
aws gamelift delete-scaling-policy \
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --name "Scale up when AGS<50"
```

Syntax für Auto Scaling-Regeln

Um eine regelbasierte Erklärung zur Skalierungsrichtlinie zu erstellen, geben Sie sechs Variablen an:

<metric name> Bleibt *<comparison operator>* *<threshold value>* für übrig *<evaluation period>*, dann ändern Sie die Flottenkapazität mit. *<adjustment type>* to/by *<adjustment value>*

Mit dieser Grundsatzerklärung wird beispielsweise ein Scale-Up-Event gestartet, wenn die zusätzliche Kapazität einer Flotte geringer ist als für 50 neue Spielsitzungen erforderlich ist:

If AvailableGameSessions remains at less than 50 for 10 minutes, then change fleet capacity using ChangeInCapacity by 1 instances.

Metrikname

Um ein Skalierungsereignis zu starten, verknüpfen Sie eine Auto Scaling-Richtlinie mit einer der folgenden flottenspezifischen Metriken. Vollständige Beschreibungen der Metriken finden Sie unter. [Amazon GameLift Servers-Metriken für Flotten](#)

- Aktivieren der Spielsitzungen
- Aktive Spielsitzungen
- Verfügbare Spielsitzungen
- Verfügbare Spielsitzungen als Prozentwert
- Aktive Instances
- Verfügbare Spiellersitzungen
- Aktuelle Spiellersitzungen
- Instances im Leerlauf
- Instances im Leerlauf als Prozentwert

Wenn sich die Flotte in einer Warteschlange für Spielsitzungen befindet, kannst du die folgenden Messwerte verwenden:

- Tiefe der Warteschlange — Die Anzahl der ausstehenden Anfragen für Spielsitzungen ist diese Flotte der beste verfügbare Hosting-Standort.
- Wartezeit — Flottenspezifische Wartezeit. Die Zeitspanne, wie lange die älteste ausstehende Spielsitzungsanfrage auf ihre Erfüllung gewartet hat. Die Wartezeit einer Flotte entspricht der Wartezeit der ältesten aktuellen Anforderung in der Warteschlange.

Vergleichsoperator

Beschreibt, Amazon GameLift Servers wie die Metrikdaten mit dem Schwellenwert verglichen werden. Zu den gültigen Vergleichsoperatoren gehören größer als (>), kleiner als (<), greater than or equal (>=) und kleiner als oder gleich (<=).

Schwellenwert

Wenn der angegebene Metrikwert den Schwellenwert erreicht oder überschreitet, wird ein Skalierungsereignis ausgelöst. Dieser Wert ist immer eine positive Ganzzahl.

Auswertungszeitraum

Die Metrik muss den Schwellenwert während des gesamten Bewertungszeitraums erreichen oder überschreiten, bevor ein Skalierungsereignis gestartet wird. Die Länge des Auswertungszeitraums ist fortlaufend; wenn die Metrik von dem Schwellenwert abweicht, beginnt der Auswertungszeitraum von neuem.

Anpassungstyp und -wert

Dieser Satz von Variablen legt zusammen fest, wie die Kapazität der Flotte zu Beginn eines Skalierungsereignisses angepasst werden Amazon GameLift Servers sollte. Wählen Sie aus drei möglichen Anpassungsarten:

- **Änderung der Kapazität** — Erhöhen oder verringern Sie die aktuelle Kapazität um eine bestimmte Anzahl von Instanzen. Legen Sie den Anpassungswert für die Anzahl der Instances fest, um den die Flotte vergrößert oder verkleinert werden soll. Positive Werte fügen Instances hinzu, während negative Werte Instances entfernen. Ein Wert von „-10“ reduziert beispielsweise die Flotte um 10 Instances, unabhängig von der Gesamtgröße der Flotte.
- **Prozentuale Änderung der Kapazität** — Erhöhen oder verringern Sie die aktuelle Kapazität um einen bestimmten Prozentsatz. Stellen Sie den Anpassungswert auf den Prozentsatz ein, um den Sie die Flottenkapazität erhöhen oder verringern möchten. Positive Werte fügen Instances hinzu, während negative Werte Instances entfernen. Beispiel: Bei einer Flotte mit 50 Instances fügt eine prozentuale Änderung von „20“ der Flotte 10 Instances hinzu.
- **Exakte Kapazität** — Erhöhen oder verringern Sie die aktuelle Kapazität auf einen bestimmten Wert. Legen Sie den Anpassungswert auf die genaue Anzahl an Instances fest, die die Flotte beibehalten soll.

Tipps für regelbasiertes Auto Scaling

Die folgenden Vorschläge können Ihnen helfen, Auto Scaling mit regelbasierten Richtlinien optimal zu nutzen.

Verwenden mehrerer Richtlinien

Sie können mehrere Auto Scaling-Richtlinien für eine Flotte gleichzeitig haben. Das häufigste Szenario ist, dass eine zielorientierte Richtlinie die meisten Skalierungsanforderungen verwaltet und regelbasierte Richtlinien verwendet, um Randfälle zu behandeln. Es gibt keine Einschränkungen bei der Verwendung mehrerer Richtlinien.

Bei mehreren Richtlinien verhält sich jede Richtlinie unabhängig. Es gibt keine Möglichkeit, die Reihenfolge der Skalierungsereignisse zu kontrollieren. Wenn Sie beispielsweise über mehrere Richtlinien verfügen, die die Skalierung vorantreiben, ist es möglich, dass Spieleraktivitäten mehrere Skalierungsereignisse gleichzeitig auslösen. Vermeiden Sie Richtlinien, die sich gegenseitig beeinflussen. Sie könnten beispielsweise eine Endlosschleife erzeugen, wenn Sie Richtlinien für das Hoch- und Herunterskalieren erstellen, bei denen die Kapazität über den jeweiligen Schwellenwert hinausgeht.

Einstellen der maximalen und minimalen Kapazität

Jede Flotte hat eine Einstellung für das maximale und für das minimale Kapazitätslimit. Diese Funktion ist wichtig, wenn Sie Auto Scaling verwenden. Auto Scaling legt die Kapazität niemals auf einen Wert fest, der außerhalb dieses Bereichs liegt. Standardmäßig haben neu erstellte Flotten einen Mindestwert von 0 und einen Höchstwert von 1. Erhöhen Sie den Maximalwert, damit sich Ihre Auto Scaling-Richtlinie wie beabsichtigt auf die Kapazität auswirkt.

Die Flottenkapazität wird auch durch Beschränkungen des Instance-Typs der Flotte und durch Servicequoten in Ihrem AWS-Konto System eingeschränkt. Außerhalb dieser Limits und Kontokontingente können Sie keine Mindest- und Höchstwerte festlegen.

Verfolgen von Metriken nach einer Kapazitätsänderung

Amazon GameLift Servers wartet nach der Änderung der Kapazität als Reaktion auf eine Auto Scaling-Richtlinie 10 Minuten, bevor auf Trigger derselben Richtlinie reagiert wird. Diese Wartezeit gibt Amazon GameLift Servers Zeit, um die neuen Instanzen hinzuzufügen, die Spielserver zu starten, Spieler zu verbinden und mit dem Sammeln von Daten von den neuen Instanzen zu beginnen. Während dieser Zeit Amazon GameLift Servers bewertet die Richtlinie anhand

der Metrik und verfolgt den Evaluierungszeitraum der Richtlinie, der nach dem Eintreten eines Skalierungsereignisses neu gestartet wird. Das bedeutet, dass eine Skalierungsrichtlinie unmittelbar nach Ablauf der Wartezeit ein weiteres Skalierungsereignis auslösen könnte.

Zwischen Skalierungsereignissen und dem Start verschiedener Auto Scaling-Richtlinien gibt es keine Wartezeiten.

Verwaltung von Scaling und Amazon GameLift Servers Fleet To/From Zero

Amazon GameLift Servers unterstützt die automatische Skalierung auf und von null Instanzen basierend auf der Aktivität der Spielsitzung. Diese Option mit verwalteter Kapazität ermöglicht es Ihren Flottenstandorten, nach einem bestimmten Zeitraum ohne Spielsitzungen auf null Instances hochzuskalieren und automatisch zu skalieren, wenn Spielsitzungen angefordert werden.

Die Skalierung auf und von Null-Instances bietet mehrere Vorteile:

- **Kostenoptimierung** — Eliminieren Sie die Rechenkosten in Zeiten der Inaktivität, indem Sie keine Instances ausführen, wenn keine Spielsitzung stattfindet.
- **Automatisches reaktives Skalieren** — Die Standorte der Flotten werden automatisch auf eine Instanz skaliert, wenn eine Spielsitzung angefordert wird, sodass kein manuelles Eingreifen erforderlich ist.
- **Vereinfachtes Management** — Es ist nicht erforderlich, die Flottenkapazität manuell auf to/from Null einzustellen, je nach erwarteter Spielernachfrage oder Entwicklungsanforderungen.

Wenn Sie Scale To/From Zero für eine Flotte aktivieren, überwacht Amazon GameLift Servers die Aktivität der Spielsitzungen und passt die Flottenkapazität automatisch an:

- **Skalierung auf Null** — Nach einem konfigurierten Zeitraum ohne Aktivität der Spielsitzungen wird der Flottenstandort auf null Amazon GameLift Servers Instances hochskaliert.
- **Skalierung von Null** — Wenn eine Anfrage zur Erstellung einer Spielsitzung eingeht, wird der Flottenstandort auf eine Instanz Amazon GameLift Servers skaliert, sodass die auto Skalierung wieder aufgenommen werden kann.
- **Kontinuierliche Skalierung** — Nach der Skalierung verwendet die Flotte wieder konfigurierte Auto-Scaling-Richtlinien zur Kapazitätsverwaltung.

Verhalten bei der Skalierung

Amazon GameLift Servers startet den Scale-In-Prozess für einen Flottenstandort nach Ablauf des konfigurierten Inaktivitätszeitraums, ohne dass eine Spielsitzung aktiv war. Dies ist definiert als ein Zeitraum, in dem:

- Am Standort Fleet gibt es keine aktiven Spielsitzungen.
- Es wurden keine Anfragen zur Erstellung neuer Spielsitzungen am Standort Fleet gestellt.

Während der Skalierung Amazon GameLift Servers werden die Mindest- und die gewünschte Kapazität für den Standort der Flotte auf Null gesetzt, wodurch eine schnelle Skalierung erzielt wird, um Kosten zu sparen.

Verhalten bei horizontaler Skalierung

Wenn eine Anfrage zur Erstellung einer Spielsitzung eingeht, während sich der Standort der Flotte auf null Instanzen befindet:

- Amazon GameLift Servers leitet sofort die Skalierung einer Instanz ein.
- Der Versuch, die Spielsitzung zu platzieren, kann je nach Konfiguration der Warteschlangen (falls verwendet) mit anderen Flotten oder Flottenstandorten fortgesetzt werden.

Note

Die Bereitstellung und Initialisierung der Instances bei einer Skalierung von Null nimmt Zeit in Anspruch. Bei Spielern kann es nach einer gewissen Zeit der Inaktivität zu längeren Wartezeiten für die erste Spielsitzung kommen. Aus diesem Grund lässt sich diese Funktion am besten mit and/or Flottenwarteschlangen an mehreren Standorten kombinieren.

Scale Zero konfigurieren To/From

Scaling To/From Zero wird konfiguriert, indem eine bestehende Flotte aktualisiert wird.

Console

1. Öffnen Sie die [Amazon GameLift Servers-Konsole](#).

2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Flottendetailseite zu öffnen.
4. Wählen Sie unter Skalierung jeden Standort aus, für den Sie Skalierung to/from Null konfigurieren möchten, und wählen Sie dann Bearbeiten aus.
5. Wählen Sie im Dialogfeld Skalierungskapazität bearbeiten die Option „Automatisch“ für die Mindestkapazitätsstrategie aus, legen Sie Ihren bevorzugten Wert in Minuten für Danach die Mindestkapazität auf 0 setzen fest und wählen Sie dann Bestätigen aus.

Amazon GameLift Servers skaliert dann die ausgewählten Flottenstandorte auf 0 Instances, sobald während der konfigurierten Dauer keine Spielsitzung stattgefunden hat. Wenn anschließend eine Anfrage für eine Spielsitzung an diesem Standort gestellt wird, Amazon GameLift Servers wird eine Instanz so schnell wie möglich skaliert. Dieser Vorgang wird einige Zeit in Anspruch nehmen.

AWS CLI

- Konfigurieren Sie die Skala to/from Null. Verwenden Sie in einem Befehlszeilenfenster den [update-fleet-capacity](#) Befehl mit der Flotten-ID, dem Standort und der verwalteten Kapazitätskonfiguration, um Scale to/from Zero zu konfigurieren.

```
aws gamelift update-fleet-capacity \
    --fleet-id <fleet identifier> \
    --location <location name> \
    --managed-capacity-configuration
    ScaleInAfterInactivityMinutes=60,ZeroCapacityStrategy=SCALE_TO_AND_FROM_ZERO
```

Beispiel:

```
aws gamelift update-fleet-capacity \
    --fleet-id
    fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
    --location us-west-2 \
    --desired-instances 5 \
    --max-size 10 \
    --managed-capacity-configuration
    ScaleInAfterInactivityMinutes=60,ZeroCapacityStrategy=SCALE_TO_AND_FROM_ZERO
```

Wenn Ihre Anfrage erfolgreich ist, wird die aktualisierte Kapazitätskonfiguration Amazon GameLift Servers zurückgegeben. Amazon GameLift Servers skaliert dann die ausgewählten Flottenstandorte auf 0 Instanzen, sobald während der konfigurierten Dauer keine Spielsitzung stattgefunden hat. Wenn anschließend eine Anfrage für eine Spielsitzung an diesem Standort gestellt wird, Amazon GameLift Servers wird eine Instanz so schnell wie möglich skaliert. Dieser Vorgang wird einige Zeit in Anspruch nehmen.

Best Practices

Beachten Sie bei der Verwendung von Scale To/From Zero die folgenden Empfehlungen:

- Legen Sie angemessene Inaktivitätszeiträume fest — Wägen Sie die Kosteneinsparungen mit der Häufigkeit von Scale-In/Scale-Out-Zyklen ab. Eine kürzere Wartezeit bis zur Skalierung auf Null könnte zwar die Einsparungen maximieren, würde aber zu häufigeren Kaltstarts führen.
- Einsatz bei vorhersehbaren Workloads — Scale To/From Zero eignet sich am besten für Spiele mit deutlichen Zeiträumen der Inaktivität, wie z. B. development/test Umgebungen oder Spiele mit ausgeprägten Randzeiten.
- Überwachen Sie CloudWatch Messwerte — Verfolgen Sie Ereignisse bei der Flottenskalierung und die Platzierungszeiten von Spielsitzungen, um Ihre Konfiguration zu optimieren.
- Kombinieren Sie mit Skalierungsrichtlinien — Verwenden Sie Scale To/From Zero zusammen mit ziel- oder regelbasierter auto Skalierung für ein umfassendes Kapazitätsmanagement.

Amazon GameLift Servers Containerflotten skalieren

Eine der schwierigsten Aufgaben beim Hosten von Spielen ist die Skalierung der Kapazität, um der Nachfrage der Spieler gerecht zu werden, ohne Geld für Ressourcen zu verschwenden, die Sie nicht benötigen. In einer verwalteten Containerflotte skalieren Sie Ihre Flottenkapazität, indem Sie Flotteninstanzen hinzufügen oder entfernen.

Wenn Sie eine neue Flotte erstellen, legen Amazon GameLift Servers Sie die gewünschte Kapazität der Flotte auf eine Instanz fest und stellen eine Instanz in der Heimatregion der Flotte bereit. Amazon GameLift Servers stellt bei einer Flotte mit mehreren Standorten eine Instanz in der Heimatregion und an jedem Remote-Standort bereit. Sobald der Flottenstatus erreicht ist `ACTIVE`, können Sie die gewünschte Kapazität erhöhen, um die gewünschte Kapazität zu erhöhen oder zu reduzieren.

Sie können Amazon GameLift Servers Skalierungsfunktionen verwenden, um die Kapazität manuell zu ändern, oder die automatische Skalierung auf der Grundlage der Spielernachfrage einrichten:

- Richten Sie die automatische Skalierung mit Zielverfolgung ein. Siehe [Zielbasiertes Auto Scaling](#).
- Ändern Sie die Kapazität Ihrer Flotte manuell. Siehe [Manuelles Einstellen der Kapazität für eine Amazon GameLift Servers Flotte](#).

Denken Sie bei der Skalierung einer Containerflotte darüber nach, wie sich das Hinzufügen oder Entfernen von Instanzen auf die Kapazität der Flotte auswirkt, Spielsitzungen und Spieler zu veranstalten.

- Spielsitzungen pro Instanz
 - Jeder Spieleserverprozess, der auf einer Instanz ausgeführt wird, entspricht der Kapazität, eine Spielsitzung zu hosten.
 - Verwenden Sie diese Formel, um die Anzahl der Spielsitzungen zu berechnen, die gleichzeitig auf einer Container-Flotteninstanz ausgeführt werden:

```
[Game sessions per instance] = [# of game server processes per game server container] * [# of game server container groups per instance]
```

Wenn Ihre Container-Architektur einen Gameserver-Prozess gleichzeitig im Gameserver-Container ausführt, dann entsprechen Spielesitzungen pro Instance der Anzahl von Gameserver-Containergruppen pro Instance.

- Rufen Sie für Spielserver-Containergruppen pro Instanz den Wert oder auf, [DescribeContainerFleet](#)um den MaximumGameServerContainerGroupsPerInstance Wert GameServerContainerGroupsPerInstance oder abzurufen.
- Spieler pro Instanz
 - Sie entscheiden, wie viele Spielerplätze in jeder Spielsitzung zulässig sind. Je nachdem, wie Ihre Hosting-Lösung die Platzierung von Spielsitzungen handhabt, können Sie in Ihrer Matchmaking-Konfiguration oder in Ihren Aufrufen zum Starten einer Spielsitzung Spieler pro Spielsitzung definieren.
 - Verwenden Sie diese Formel, um die Anzahl der Spieler zu berechnen, die Ihr Spiel gleichzeitig auf einer Container-Flotteninstanz spielen können:

```
[Players per instance] = [# of game sessions per instance] * [# of player slots per game session]
```

Um die aktuelle Gesamtkapazität einer Containerflotte zu ermitteln, rufen Sie [DescribeFleetCapacityDescribeFleetLocation Capacity](#) auf, um die Anzahl der Spielserver-Containergruppen in der Flotte abzurufen. Aktive Gruppen sind Gruppen, die derzeit Spielsitzungen veranstalten. Inaktive Gruppen sind bereit, eine neue Spielsitzung zu veranstalten. Multiplizieren Sie diese Werte mit der Anzahl der Serverprozesse pro Spielserver-Containergruppe.

Überwachung von Amazon GameLift Servers

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer Amazon GameLift Servers anderen AWS Lösungen. Es gibt in Amazon GameLift Servers drei Hauptnutzungsmöglichkeiten für Metriken: zur Überwachung des Systemzustands und zum Einrichten von Alarmen, zur Nachverfolgung der Leistung und der Auslastung von Spiel-Servern und zur Verwaltung der Kapazität mit manueller oder automatischer Skalierung.

AWS bietet die folgenden Überwachungstools, mit denen Sie beobachten Amazon GameLift Servers, melden können, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen ergreifen können:

- **Amazon GameLift Servers Konsole** — Verwende die grafische Oberfläche, um deine Amazon GameLift Servers Ressourcen zu verwalten und die Aktivitäten beim Hosten von Spielen zu verfolgen.
- **Server-Telemetrie-Metriken** — Amazon GameLift Servers ermöglicht es dir, benutzerdefinierte Messwerte direkt von deinen Spieleservern über SDK- und Plugin-Integration auszugeben. Du kannst deine eigenen spielspezifischen Messwerte zusammen mit integrierten Leistungs-, Netzwerk-, Speicher- und Timing-Daten definieren und verfolgen. Alle Metriken können in [Amazon Managed Service for Prometheus](#) veröffentlicht und in Amazon Grafana mithilfe vollständig anpassbarer Dashboards überwacht werden. Sie können auch auf Amazon CloudWatch veröffentlichen, um sie in andere AWS Dienste zu integrieren. Das Telemetriesystem ist vollständig anpassbar — Sie können benutzerdefinierte Prometheus-Abfragen erstellen, um zusätzlich zu den integrierten Metriken zusätzliche Metriken abzuleiten. Implementierungsleitfäden speziell für Ihren Technologie-Stack finden Sie unter [the section called “Überwachung mit Server-Telemetrie-Metriken”](#)
- **Amazon CloudWatch** — Sie können Amazon GameLift Servers Metriken in Echtzeit überwachen, ebenso wie Metriken für andere AWS Ressourcen und Anwendungen, die Sie auf AWS Diensten ausführen. Amazon CloudWatch bietet eine Reihe von Überwachungsfunktionen, darunter Tools zur Erstellung benutzerdefinierter Dashboards und die Möglichkeit, Alarme einzurichten, die benachrichtigen oder Maßnahmen ergreifen, wenn eine Metrik einen bestimmten Schwellenwert erreicht.
- **AWS CloudTrail** — erfasst alle API-Aufrufe und damit verbundene Ereignisse, die von oder im Namen Ihres AWS Kontos Amazon GameLift Servers und anderer AWS Dienste getätigt wurden. Daten werden als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket

übermittelt. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Anrufe erfolgten.

- Spielsitzungsprotokolle — Sie können benutzerdefinierte Servermeldungen für Ihre Spielsitzungen in Protokolldateien ausgeben, die in Amazon S3 gespeichert sind.

Vergleich der Messwerte verschiedener Überwachungsquellen

Amazon GameLift Servers stellt Metriken über drei Hauptquellen bereit: die Flottenaktivitätsmetriken der Amazon GameLift Servers Konsole, Server-Telemetrie-Metriken und CloudWatch Amazon GameLift Servers Amazon-Metriken. Wenn Sie die Überschneidungen und die einzigartigen Funktionen der einzelnen Quellen verstehen, können Sie den richtigen Überwachungsansatz für Ihre Bedürfnisse auswählen.

Verfügbarkeit von Metriken nach Quelle

Die folgenden Tabellen zeigen, welche Metriken in den verschiedenen Monitoring-Quellen verfügbar sind, geordnet nach Metrikkategorien.

Instance-Metriken

Metriken auf Instanzebene für die Überwachung der Flottenkapazität und des Zustands:

Verfügbarkeit von Instanzmetriken

Metrik	Konsole	CloudWatch	Telemetrie
Aktive Instances	✓	✓	✓
Instances im Leerlauf	✓	✓	✓*
Instances im Leerlauf als Prozentwert	✓	✓	✓*
Gewünschte Instanzen	✓	✓	
Max. Anzahl Instanzen	✓	✓	
Wichtigste Instanzen	✓	✓	
CPU-Auslastung		✓	✓

Metrik	Konsole	CloudWatch	Telemetrie
Netzwerk ein-/ausgehen		✓	✓
Disk/Storage read/write		✓	✓
Unterbrechungen der Instanz	✓	✓	
Wiederverwertete Instanzen	✓	✓	
Fehlerhafte Instanzen wurden ersetzt		✓	

* Verfügbar über benutzerdefinierte Prometheus-Abfragen unter Verwendung von Telemetriedaten.

Metriken für Spieleserver

Metriken zu Serverprozessen und Spielsitzungen:

Verfügbarkeit von Spielserver-Metriken

Metrik	Konsole	CloudWatch	Telemetrie
Aktive Spielsitzungen	✓	✓	✓*
Aktivieren der Spielsitzungen	✓	✓	
Verfügbare Spielsitzungen	✓	✓	
Verfügbare Spielsitzungen als Prozentwert	✓	✓	
Gleichzeitige aktivierbare Spielsitzungen	✓	✓	
Unterbrechungen der Spielsitzungen	✓	✓	
Aktive Serverprozesse			✓
Gesunde Spieleserver			✓
Abgestürzte Spielsitzungen			✓

* Verfügbar über benutzerdefinierte Prometheus-Abfragen unter Verwendung von Telemetriedaten.

Metriken zur Spielernutzung

Statistiken zu Spielersitzungen und gleichzeitigen Benutzern:

Verfügbarkeit von Nutzungsmetriken für Spieler

Metrik	Konsole	CloudWatch	Telemetrie
Aktuelle Spielersitzungen	✓		
Aktivierungen von Spielersitzungen	✓		
Globale gleichzeitige Benutzer (CCU)			✓
Standort für gleichzeitige Benutzer (CCU)			✓
Kapazität des Standorts			✓

Kennzahlen zur Containerflotte

Containerspezifische Kennzahlen (nur CloudWatch für Containerflotten verfügbar):

Metriken zur Containerflotte: Verfügbarkeit

Metrik	Konsole	CloudWatch	Telemetrie
Aktive Containergruppen		✓	✓
Containergruppen im Leerlauf		✓	✓
CPU/Memory Nutzung von Containern		✓	✓
Container-Netzwerkverkehr		✓	✓
Betrieb der Lagerung von Containern		✓	✓

Auswahl der richtigen Überwachungsquelle

Wählen Sie Ihren Überwachungsansatz auf der Grundlage Ihrer spezifischen Bedürfnisse aus:

- Verwenden Sie Servertelemetriemetriemetriemetrien, um umfassende Leistungsdaten von Ihren Spieleservern zu überwachen und benutzerdefinierte Messwerte für Ihr Spiel auszugeben. Diese Messwerte liefern detaillierte Einblicke in die Serverleistung, Netzwerkaktivität, Speicherauslastung und Timing-Daten für alle Game-Engines und Server SDKs. Sie können benutzerdefinierte Messwerte für Gameplay-Ereignisse, die Leistung der Geschäftslogik und anwendungsspezifische Datenpunkte definieren. Alle Dashboards sind vollständig anpassbar, und Sie können benutzerdefinierte Prometheus-Abfragen erstellen, um zusätzliche Metriken aus den gesammelten Daten abzuleiten. Weitere Informationen finden Sie unter [the section called “Überwachung mit Server-Telemetrie-Metriemetriemetri”](#).
- Verwenden Sie die Amazon GameLift Servers Konsole für das Flottenmanagement, die Kapazitätsplanung und die allgemeine Betriebsüberwachung. Die Konsole bietet einen integrierten Überblick über den Zustand der Flotte und die Spieleraktivitäten. Weitere Informationen finden Sie unter [the section called “Ressourcen zum Hosten von Spielen verwalten”](#).
- Verwenden Sie Amazon CloudWatch für automatisierte Überwachung, Warnmeldungen und Integration mit anderen AWS Diensten. CloudWatch ermöglicht benutzerdefinierte Dashboards und alarmbasierte Automatisierung. Weitere Informationen finden Sie unter [the section called “Überwachen Sie mit CloudWatch”](#).
- Verwenden Sie [Amazon Managed Service for Prometheus](#) für die leistungsstarke Erfassung und Speicherung von Kennzahlen mit PromQL-Abfragefunktionen. Prometheus bietet skalierbaren Zeitreihendatenspeicher für Server-Telemetriemetriemetriemetriken.
- Verwenden Sie [Amazon Managed Grafana](#) für erweiterte Visualisierungen und vollständig anpassbares Dashboarding. Grafana bietet vorgefertigte GameLift Dashboards, die Sie anpassen und erweitern können, und unterstützt außerdem die Erstellung vollständig benutzerdefinierter Dashboards mit mehreren Datenquellen. Sie können benutzerdefinierte Abfragen und Visualisierungen erstellen, um alle für Ihr Spiel wichtigen Metriken zu verfolgen.
- Verwenden Sie mehrere Quellen für eine umfassende Überwachung. Kombinieren Sie Konsolenüberwachung, Servertelemetriemetriemetriken und automatisierte Benachrichtigungen, um einen vollständigen Überblick über Ihre Game-Hosting-Infrastruktur zu erhalten.

Für zusätzliche Überwachungsfunktionen können Sie auch Folgendes verwenden:

- [the section called “Protokollieren von API-Aufrufen”](#)— Verfolgen Sie API-Aufrufe und damit verbundene Ereignisse zur Prüfung und Einhaltung von Vorschriften.
- [the section called “Servernachrichten protokollieren”](#)— Erfassen Sie benutzerdefinierte Servernachrichten und Spielsitzungsprotokolle.

Topics

- [the section called “Ressourcen zum Hosten von Spielen verwalten”](#)
- [the section called “Überwachen Sie mit CloudWatch”](#)
- [the section called “Überwachung mit Server-Telemetrie-Metriemetriemetri”](#)
- [the section called “Protokollieren von API-Aufrufen”](#)
- [the section called “Servernachrichten protokollieren”](#)

Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch

Sie können die Überwachung Amazon GameLift Servers mithilfe von Amazon durchführen CloudWatch, einem AWS Service, der Rohdaten sammelt und sie zu lesbaren Metriken verarbeitet, die nahezu in Echtzeit verfügbar sind. Diese Statistiken werden 15 Monate lang aufbewahrt, um eine historische Perspektive zu bieten, wie Ihr Spielserver-Hosting mit Amazon GameLift Servers funktioniert. Sie können auch Alarme einrichten, die auf bestimmte Grenzwerte prüfen und Benachrichtigungen senden oder Aktivitäten auslösen, wenn diese Grenzwerte erreicht werden. Weitere Informationen finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).

[Für eine verbesserte serverseitige Beobachtbarkeit mit detaillierten Telemetrie-Metriken wird Amazon GameLift Servers auch die Überwachung über Amazon Managed Service for Prometheus und Amazon Managed Grafana unterstützt.](#) Diese Telemetrie-Metriken liefern detaillierte Serverleistungsdaten, darunter:

- Server-Timings: Server Delta Time (Konsistenz der Server-Tick-Rate), Server Tick Rate (Updates pro Sekunde), Server Tick Time (Verarbeitungszeit pro Tick) und Server World Tick Time (Spielwelt-Aktualisierungszeit)
- Netzwerkmetriken: Verbindungen (Gesamtzahl der Netzwerkverbindungen), Netzwerk I/O in Byte und Paketen (Datenübertragungsstatistiken) und Paketverlust (Prozentsatz der Übertragungsfehler)
- Speichermetriken: Speicherauslastung in Einheiten (Gesamtspeicherverbrauch) und Physikalische Speicherauslastung in Prozent (Speicherauslastung)
- CPU-Auslastung: Prozentsatz der CPU-Auslastung (Ressourcennutzung durch den Spieleserverprozess)

Implementierungsleitfäden speziell für Ihr SDK oder Plugin finden Sie unter [the section called “Überwachung mit Server-Telemetrie-Metriemetriemetri”](#).

Die folgende Tabelle listet die Metriken und Dimensionen für Amazon GameLift Servers auf. Alle Metriken, die in verfügbar CloudWatch sind, sind auch in der Amazon GameLift Servers Konsole verfügbar, die die Daten in Form von anpassbaren Diagrammen bereitstellt. Verwenden Sie die, die oder die AWS-Managementkonsole CloudWatch API, um auf CloudWatch Metriken für Ihre Spiele zuzugreifen. AWS CLI

Wenn eine Metrik keinen Standort hat, verwendet sie den Heimatstandort.

Dimensionen für Amazon GameLift Servers-Metriken

Amazon GameLift Servers unterstützt die Filterung von Metriken nach den folgenden Dimensionen.

Dimension	Description
Location	Filtern Sie die Metriken nach einem Einsatzort für Ihre Flotte. Wenn eine Metrik keinen Standort hat, verwendet sie den Heimatstandort.
FleetId	Filtermetriken für eine einzelne Flotte. Diese Dimension kann für alle Instance-, Serverprozess-, Spielsitzungs- und Spielersitzungsmetriken verwendet werden.
MetricGroup	Filtermetriken für eine Sammlung von Flotten. Fügen Sie einer Metrikgruppe eine Flotte hinzu, indem Sie den Namen der Metrikgruppe zu den Flottenattributen hinzufügen UpdateFleetAttributes(siehe ()) . Diese Dimension kann für alle Instance-, Serverprozess-, Spielsitzungs- und Spielersitzungsmetriken verwendet werden.
QueueName	Filtermetriken für eine einzelne Warteschlange. Die Dimension wird nur für Metriken für Spielsitzungswarteschlangen verwendet.

Dimension	Description
ConfigurationName	Filtermetriken für eine einzelne Matchmaking-Konfiguration. Diese Dimension wird für Metriken für Matchmaking-Konfigurationen verwendet.
ConfigurationName-RuleName	Filtermetriken für einen Schnittpunkt einer Matchmaking-Konfiguration und Matchmaking-Regel. Diese Dimension wird nur für Metriken für Matchmaking-Regeln verwendet.
InstanceType	Filtermetriken für eine EC2-Instance-Typ-Zuordnung (z. B. „c4.large2“). Diese Dimension wird für Spot-Instance-Metriken verwendet.
OperatingSystem	Filtermetriken für das Betriebssystem einer Instance. Diese Dimension wird mit Metriken für Spot-Instances verwendet.
GameServerGroup	Filter-FleetIQ-Metriken für eine Spielservergruppe.
ContainerGroupType	Filtern Sie die Kennzahlen für die Containerflotte, um zwischen GameServer ContainerGroup Kennzahlen zum Flottenstandort und PerInstance Container Group Kennzahlen zum Flottenstandort zu unterscheiden. Diese Dimension ist für bestimmte Kennzahlen zur Containerflotte (z. B. ContainerStorageReadBytes) enthaltenContainerNetworkIn , sodass Sie zwischen Kennzahlen für verschiedene Containergruppentypen unterscheiden können.

Amazon GameLift Servers-Metriken für Flotten

Der Amazon GameLift-Namespace enthält die folgenden Metriken hinsichtlich der Aktivitäten einer Flotte oder Flottengruppe. Flotten werden mit einer verwalteten Amazon GameLift Servers-Lösung verwendet. Der Amazon GameLift Servers Service sendet Metriken an CloudWatch jede Minute.

Instances

Metrik	Description
ActiveInstances	Instances mit dem Status ACTIVE, d. h. solche, auf denen aktive Serverprozesse ausgeführt werden. Die Anzahl umfasst die nicht verwendeten Instances sowie jene, die mehr als eine Spielsitzung hosten. Diese Metrik misst die aktuelle Gesamtkapazität der Instance. Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
DesiredInstances	Zielanzahl der aktiven Instances, die Amazon GameLift Servers betreibt, um die Flotte zu erhalten. Mit Auto Scaling wird dieser Wert anhand der aktuell gültigen Skalierungsrichtlinien bestimmt. Ohne Auto Scaling wird dieser Wert manuell festgelegt. Diese Metrik ist nicht verfügbar, wenn Metrik-Daten für Flottengruppen angezeigt werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
IdleInstances	Aktive Instances, die gegenwärtig null (0) Spielsitzungen hosten. Diese Metrik misst die verfügbare, aber nicht verwendete Kapazität. Diese Metrik kann mit Auto Scaling verwendet werden.

Metrik	Description
	Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
MaxInstances	Maximale Anzahl der Instances, die für die Flotte zugelassen sind. Der Maximalwert einer Flotten-Instance bestimmt die Kapazitätsgrenze während der manuellen oder automatischen Aufwärtsskalierung. Diese Metrik ist nicht verfügbar, wenn Metrik-Daten für Flottengruppen angezeigt werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
MinInstances	Minimale Anzahl der Instances, die für die Flotte zugelassen sind. Der Minimalwert einer Flotten-Instance bestimmt die Kapazitätsgrenze während der manuellen oder automatischen Abwärtsskalierung. Diese Metrik ist nicht verfügbar, wenn Metrik-Daten für Flottengruppen angezeigt werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Description
PercentIdleInstances	Prozentsatz der ungenutzten aktiven Instances (berechnet als $\text{IdleInstances} / \text{ActiveInstances}$). Diese Metrik kann für Auto Scaling verwendet werden. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
RecycledInstances	Anzahl der Spot-Instances, die recycelt und ersetzt wurden. Amazon GameLift Servers recycelt Spot-Instances, die derzeit keine Spielsitzungen hosten und bei denen die Wahrscheinlichkeit einer Unterbrechung hoch ist. Diese Metrik ist nur für verwaltete EC2-Flotten verfügbar. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Description
RecycledSpotInstances	Anzahl der Spot-Instances, die recycelt und ersetzt wurden. Amazon GameLift Servers recycelt Spot-Instances, die derzeit keine Spielsitzungen hosten und bei denen die Wahrscheinlichkeit einer Unterbrechung hoch ist. Diese Metrik ist nur für Containerflotten verfügbar. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort
InstanceSpotInterruptions	Anzahl der Spot-Instances, die unterbrochen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Description
CPUUtilization	Metrisch EC2. Denn Amazon GameLift Servers diese Metrik stellt die Hardwareleistung aller aktiven Instances an einem Flottenstandort dar. Der Prozentsatz der physischen CPU-Zeit, die Amazon EC2 zum Ausführen der Instance verwendet , einschließlich der Zeit, die sowohl für die Ausführung des Benutzercodes als auch für den Amazon EC2 EC2-Code aufgewendet wurde. Tools in Ihrem Betriebssystem können CloudWatch aufgrund von Faktoren wie Legacy-Gerätesimulation, Konfiguration nicht älterer Geräte, Interrupt-intensive Workloads , Live-Migration und Live-Update einen anderen Prozentsatz anzeigen. Einheiten: Prozent
NetworkIn	EC2-Metrik. Denn Amazon GameLift Servers diese Metrik stellt die Hardwareleistung aller aktiven Instances an einem Flottenstandort dar. Anzahl der von der Instance auf allen Netzwerkschnittstellen empfangenen Byte. Diese Metrik gibt das an eine Anwendung auf einer einzelnen Instance eingehende Netzwerkdatenvolumen an. Einheiten: Byte
NetworkOut	EC2-Metrik. Denn Amazon GameLift Servers diese Metrik stellt die Hardwareleistung aller aktiven Instances an einem Flottenstandort dar. Anzahl der von der Instance auf allen Netzwerkschnittstellen gesendeten Byte. Diese Metrik gibt das an eine Anwendung auf einer einzelnen Instance ausgehende Netzwerkdatenvolumen an. Einheiten: Byte

Metrik	Description
DiskReadBytes	EC2-Metrik. Denn Amazon GameLift Servers diese Metrik stellt die Hardwareleistung aller aktiven Instances an einem Flottenstandort dar. Von allen der Instance zu Verfügung stehenden Instance-Speicher-Volumes gelesene Byte. Diese Metrik wird verwendet, um das von der Festplatte der Instance gelesene Datenvolumen der Anwendung zu ermitteln. Sie können damit die Geschwindigkeit der Anwendung bestimmen. Einheiten: Byte
DiskWriteBytes	EC2-Metrik. Denn Amazon GameLift Servers diese Metrik stellt die Hardwareleistung aller aktiven Instances an einem Flottenstandort dar. In alle der Instance zu Verfügung stehenden Instance-Speicher-Volumes geschriebene Byte. Diese Metrik wird verwendet, um das auf die Festplatte der Instance geschriebene Datenvolumen der Anwendung zu ermitteln. Sie können damit die Geschwindigkeit der Anwendung bestimmen. Einheiten: Byte
DiskReadOps	EC2-Metrik. Denn Amazon GameLift Servers diese Metrik stellt die Hardwareleistung aller aktiven Instances an einem Flottenstandort dar. Abgeschlossene Lesevorgänge von allen der Instance zu Verfügung stehenden Instance-Speicher-Volumes in einem angegebenen Zeitraum. Um die durchschnittlichen I/O Operationen pro Sekunde (IOPS) für den Zeitraum zu berechnen, dividieren Sie die Gesamtzahl der Vorgänge in diesem Zeitraum durch die Anzahl der Sekunden in diesem Zeitraum. Einheiten: Anzahl

Metrik	Description
DiskWriteOps	EC2-Metrik. Denn Amazon GameLift Servers diese Metrik stellt die Hardwareleistung aller aktiven Instances an einem Flottenstandort dar. Abgeschlossene Schreibvorgänge von allen der Instance zu Verfügung stehenden Instance-Speicher-Volumes in einem angegebenen Zeitraum. Um die durchschnittlichen I/O Operationen pro Sekunde (IOPS) für den Zeitraum zu berechnen, dividieren Sie die Gesamtzahl der Vorgänge in diesem Zeitraum durch die Anzahl der Sekunden in diesem Zeitraum. Einheiten: Anzahl

Serverprozesse

Metrik	Description
ActiveServerProcesses	Serverprozesse mit dem Status ACTIVE, d. h. solche, die laufen und Spielsitzungen hosten können. Die Anzahl umfasst die ungenutzten Serverprozesse und jene, die Spielsitzungen hosten. Diese Metrik misst die aktuelle Gesamt-Serverprozesskapazität. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
HealthyServerProcesses	Aktive stabile Serverprozesse. Diese Metrik ist beim Nachverfolgen des übergreifenden Zustands der Spielserver der Flotte nützlich. Einheiten: Anzahl

Metrik	Description
	Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
PercentHealthyServerProcesses	Prozentsatz aller aktiven stabilen Serverprozesse (berechnet als <code>HealthyServerProcesses / ActiveServerProcesses</code>). Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
ServerProcessAbnormalTerminations	Serverprozesse, die seit dem letzten Bericht infolge anormaler Umstände heruntergefahren wurden. Diese Metrik umfasst Beendigungen, die vom Amazon GameLift Servers-Service initiiert wurden. Dies tritt auf, wenn ein Serverprozess nicht mehr reagiert, regelmäßig fehlgeschlagene Integritätsprüfungen meldet oder nicht ordnungsgemäß beendet wird (durch Aufrufen von ProcessEnding()). Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Description
ServerProcessActivations	Serverprozesse, die seit dem letzten Bericht erfolgreich vom Status ACTIVATING in den Status ACTIVE gewechselt sind. Um Spielsitzungen hosten zu können, müssen die Serverprozesse aktiv sein. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort
ServerProcessTerminations	Serverprozesse, die seit dem letzten Bericht heruntergefahren wurden. Dies umfasst alle Serverprozesse, die aus irgendeinem Grund in den Status TERMINATED gewechselt sind, einschließlich ordnungsgemäße und nicht ordnungsgemäße Prozessbeendigungen. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Spielsitzungen

Metrik	Description
ActivatingGameSessions	Spielsitzungen mit dem Status ACTIVATING, d. h. solche, die sich im Hochfahrprozess befinden. Spielsitzungen können Spieler erst dann hosten, wenn sie aktiv sind. Über einen längeren Zeitraum anliegende hohe Werte weisen darauf hin, dass die Spielsitzungen nicht vom Status ACTIVATING in den

Metrik	Description
	Status ACTIVE wechseln. Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
ActiveGameSessions	Spielsitzungen mit dem Status ACTIVE, d. h. solche, die Spieler hosten können und null oder mehr Spieler hosten. Diese Metrik misst die Gesamtzahl der Spielsitzungen, die gegenwärtig gehostet werden. Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Description
AvailableGameSessions	Aktive, fehlerfreie Serverprozesse, die derzeit nicht zum Hosten einer Spielsitzung verwendet werden und ohne Verzögerung eine neue Spielsitzung starten können, um neue Serverprozesse oder Instanzen hochzufahren. Diese Metrik kann mit Auto Scaling verwendet werden.  Note Verwenden Sie für Flotten, die die Aktivierung gleichzeitiger Spielsitzungen einschränken, die Metrik <code>ConcurrentActiveGameSessions</code>. Diese Metrik stellt die Anzahl neuer Spielsitzungen, die ohne jegliche Verzögerung gestartet werden können, genauer dar. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Description
ConcurrentActivatableGameSessions	Aktive, fehlerfreie Serverprozesse, die derzeit nicht zum Hosten einer Spielsitzung verwendet werden und sofort eine neue Spielsitzung starten können. Diese Metrik unterscheidet sich AvailableGameSessions in folgender Hinsicht: Sie zählt keine Serverprozesse, die aufgrund von Beschränkungen bei der Aktivierung von Spielsitzungen derzeit keine neue Spielsitzung aktivieren können. (Weitere Informationen finden Sie in der RuntimeConfiguration optionalen Einstellung „Flotte“ <code>MaxConcurrentGameSessionActivations</code>). Für Flotten, die die Aktivierungen von Spielsitzungen nicht einschränken, ist diese Metrik identisch mit AvailableGameSessions Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
PercentAvailableGameSessions	Spielsitzungs-Slots auf allen aktiven (stabilen oder instabilen) Servern, die gegenwärtig nicht verwendet werden (berechnet als $\text{AvailableGameSessions} / [\text{ActiveGameSessions} + \text{AvailableGameSessions} + \text{unhealthy server processes}]$). Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt Abmessungen: Standort

Metrik	Description
GameSessionInterruptions	Anzahl der Spielsitzungen auf Spot-Instances, die unterbrochen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Spiellersitzungen

Metrik	Description
CurrentPlayerSessions	Spiellersitzungen, die sich im Status ACTIVE (der Spieler ist mit einer aktiven Spielsitzung verbunden) oder RESERVED (dem Spieler ist ein Slot in einer Spielsitzung zugewiesen worden, hat sich aber noch nicht verbunden) befinden. Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum
PlayerSessionActivations	Spiellersitzungen, die seit dem letzten Bericht vom Status RESERVED in den Status ACTIVE gewechselt sind. Dies tritt ein, wenn sich ein Spieler erfolgreich mit einer aktiven Spielsitzung verbindet. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum

Kennzahlen zur Containerflotte

Kennzahlen zur Containerflotte liefern Überwachungsdaten für das Container-basierte Spiele-Hosting, darunter den Status von Containergruppen, die Ressourcennutzung und Leistungskennzahlen.

Note

Verbesserte Dashboards für die Überwachung von Containerflotten sind über [Amazon Managed Grafana](#) verfügbar, wenn Sie die Lösung für Telemetriemetriken verwenden. Anweisungen [the section called “Überwachung mit Server-Telemetrie-Metriemetriemetri”](#) zur Einrichtung finden Sie unter.

Metrik	Description
ActiveGameServerContainerGroups	Container-Gruppen, die derzeit aktiv und bereit sind, Spielsitzungen abzuhalten. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
IdleGameServerContainerGroups	Container-Gruppen, die aktiv sind, aber derzeit keine Spielsitzungen veranstalten. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
PendingGameServerContainerGroups	Containergruppen, die gerade gestartet werden. Einheiten: Anzahl

Metrik	Description
	Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
TerminatingGameServerContainerGroups	Containergruppen, die gerade heruntergefahren werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
UnhealthyGameServerContainerGroupsReplaced	Containergruppen, die aufgrund fehlgeschlagener Integritätsprüfungen ersetzt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort
ContainerCPUUtilizationPerInstance	Durchschnittliche CPU-Auslastung aller Container auf einer Instance. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Description
ContainerCPUReservation	Prozentsatz der CPU-Ressourcen, die von Containern reserviert wurden. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
ContainerMemoryReservation	Prozentsatz der von Containern reservierten Speicherressourcen. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
ContainerMemoryUtilization	Prozentsatz des reservierten Speichers, der von Containern genutzt wird. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
ContainerNetworkIn	Von Containern empfangene Netzwerk-Bytes. Einheiten: Byte Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Description
ContainerNetworkOut	Von Containern gesendete Netzwerk-Bytes. Einheiten: Byte Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort
ContainerStorageReadBytes	Von Containern aus dem Speicher gelesene Bytes. Einheiten: Byte Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort
ContainerStorageWriteBytes	Von Containern in den Speicher geschriebene Bytes. Einheiten: Byte Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

DDoMetriken zum S-Schutz (Spieler-Gateway)

Amazon GameLift ServersPlayer-Gateway-Metriken verfolgen den UDP-Verkehr, der durch die DDo S-Schutzschicht fließt, wenn auf `ENABLED` oder gesetzt `PlayerGatewayMode` ist `REQUIRED`. Diese Metriken helfen Ihnen dabei, Verkehrsmuster zu überwachen, potenzielle DDo S-Angriffe zu identifizieren und die Relay-Leistung zu verfolgen.

Metrik	Description
PlayerGatewayPacketsIn	Anzahl der UDP-Pakete, die von Spielern empfangen und über die DDo S-Schutzschicht an Spieleserver weitergeleitet wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: FleetId, Standort
PlayerGatewayBytesIn	Anzahl der Byte, die von Spielern empfangen und über die DDo S-Schutzschicht an Spieleserver weitergeleitet wurden. Einheiten: Byte Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: FleetId, Standort
PlayerGatewayPacketsOut	Anzahl der UDP-Pakete, die von Spieleservern empfangen und über die DDo S-Schutzschicht an Spieler weitergeleitet wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: FleetId, Standort
PlayerGatewayBytesOut	Anzahl der Byte, die von Spieleservern empfangen und über die DDo S-Schutzschicht an Spieler weitergeleitet wurden. Einheiten: Byte

Metrik	Description
	Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: FleetId, Standort
PlayerGatewayPacketsThrottled	Anzahl der eingehenden Pakete, die aufgrund der Ratenbegrenzung durch die DDoS-Schutzschicht verworfen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: FleetId, Standort
PlayerGatewayBytesThrottled	Anzahl der eingehenden Byte, die aufgrund der Ratenbegrenzung durch die DDoS-Schutzschicht verloren gegangen sind. Einheiten: Byte Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: FleetId, Standort
PlayerGatewayPlayerSessions	Spieler Sitzungen, die derzeit durch die DDoS-Schutzschicht geschützt sind. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: FleetId, Standort

Amazon GameLift Servers-Metriken für Warteschlangen

Der Amazon GameLift-Namespace enthält die folgenden Metriken hinsichtlich der Aktivitäten in einer Spielsitzungsplatzierungs-Warteschlange. Warteschlangen werden mit einer verwalteten Amazon GameLift Servers-Lösung verwendet. Der Amazon GameLift Servers Service sendet Metriken in CloudWatch jede Minute.

Metrik	Description
AverageWaitTime	Durchschnittliche Zeit, die sich die Anforderungen zur Spielsitzungsplatzierung in der Warteschlange mit dem Status PENDING befunden haben, bis sie erfüllt worden sind. Einheiten: Sekunden Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe Abmessungen: Standort
FirstChoiceNotViable	Spielsitzungen, die erfolgreich platziert wurden, jedoch NICHT in der Flotte der ersten Wahl, da diese Flotte als nicht geeignet betrachtet wurde (da es sich beispielsweise um eine Spot-Flotte mit einer hohen Unterbrechungsrate handelt). Diese Kennzahl basiert auf den Kosten, nicht auf der Latenz. Die Flotte erster Wahl ist entweder die erste Flotte, die in der Warteschlange aufgeführt ist, oder — wenn eine Platzierungsanfrage Latenzdaten für Spieler enthält — ist sie die erste Flotte, die nach Priorisierung ausgewählt wird. FleetIQ Wenn es keine realisierbare Spot-Flotten gibt, kann eine beliebige Flotte in dieser Region ausgewählt werden. Einheiten: Anzahl

Metrik	Description
	Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
FirstChoiceOutOfCapacity	Spielesitzungen, die erfolgreich platziert wurden, jedoch NICHT in der Flotte der ersten Wahl, da in dieser Flotte keine Ressourcen verfügbar waren. Die Flotte erster Wahl ist entweder die erste Flotte, die in der Warteschlange aufgeführt ist, oder — wenn eine Platzierungsanfrage Latenzdaten für Spieler enthält — ist sie die erste Flotte, die anhand Ihrer festgelegten Priorisierung ausgewählt wird. FleetIQ Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
LowestLatencyPlacement	Spielesitzungen, die erfolgreich in einer Region platziert wurden, die den Spielern die niedrigste Latenz in der Warteschlange bietet, die möglich ist. Diese Metrik wird nur ausgegeben, wenn in der Platzierungsanforderung Spielerlatenzdaten enthalten sind. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe

Metrik	Description
LowestPricePlacement	Spielsitzungen, die erfolgreich in eine Flotte mit dem niedrigstmöglichen Preis der Warteschlange für die gewählte Region aufgenommen wurden. Diese Flotte kann entweder eine Spot-Flotte oder eine On-Demand-Instance sein, wenn die Warteschlange keine Spot-Instances enthält. Diese Metrik wird nur ausgegeben, wenn in der Platzierungsanforderung Spielerlatenzdaten enthalten sind. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
Placement <region name>	Spielsitzungen, die erfolgreich in Flotten platziert wurden, die sich in der angegebenen Region befinden. Diese Metrik analysiert die PlacementSucceeded -Metrik nach Region. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
PlacementsCanceled	Anforderungen für Spielsitzungsplatzierungen, die seit dem letzten Bericht vor Erreichen des Zeitlimits abgebrochen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe

Metrik	Description
PlacementsFailed	Nicht erfolgreiche Anforderungen für Spielsitzungsplatzierungen seit dem letzten Bericht. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
PlacementsStarted	Neue Anforderungen für Spielsitzungsplatzierungen, die seit dem letzten Bericht der Warteschlange hinzugefügt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
PlacementsSucceeded	Neue Anforderungen für Spielsitzungsplatzierungen, die seit dem letzten Bericht zu einer neuen Spielsitzung geführt haben. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
PlacementsTimedOut	Anforderungen für Spielsitzungsplatzierungen, die seit dem letzten Bericht das Zeitlimit der Warteschlange erreicht haben, ohne erfüllt worden zu sein. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe

Metrik	Description
QueueDepth	Anzahl der Anforderungen für Spielsitzungsplatzierungen in der Warteschlange mit dem Status PENDING. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe Abmessungen: Standort

Amazon GameLift Servers-Metriken für das Matchmaking

Der Amazon GameLift-Namespace enthält Metriken zur FlexMatch-Aktivität für Matchmaking-Konfigurationen und Matchmaking-Regeln. FlexMatch-Matchmaking wird mit einer verwalteten Amazon GameLift Servers-Lösung verwendet. Der Amazon GameLift Servers Dienst sendet Metriken an CloudWatch jede Minute.

Weitere Informationen zur Reihenfolge der Matchmaking-Aktivitäten findest du unter [So Amazon GameLift Servers FlexMatch funktioniert es](#).

Matchmaking-Konfigurationen

Metrik	Description
CurrentTickets	Aktuell verarbeitete oder auf die Verarbeitung wartende Matchmaking-Anforderungen. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
MatchAcceptancesTimedOut	Für Matchmaking-Konfigurationen, die eine Akzeptanz bedingen, die potenziellen Matches, für die seit dem letzten Bericht während des Akzeptierungsvorgangs ein Timeout aufgetreten ist.

Metrik	Description
	Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
MatchesAccepted	Für Matchmaking-Konfigurationen, die eine Akzeptanz bedingen, die seit dem letzten Bericht akzeptierten potenziellen Matches. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
MatchesCreated	Potenzielle Matches, die seit dem letzten Bericht erstellt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
MatchesPlaced	Matches, die seit dem letzten Bericht in einer Spielsitzung platziert wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
MatchesRejected	Für Matchmaking-Konfigurationen, die eine Akzeptanz bedingen, die potenziellen Matches, für die seit dem letzten Bericht von mindestens einem Spieler eine Ablehnung erteilt wurde. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe

Metrik	Description
PlayersStarted	Spieler in Matchmaking-Tickets, die seit dem letzten Bericht hinzugefügt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
TicketsFailed	Matchmaking-Anforderungen, die seit dem letzten Bericht zu einem Fehler geführt haben. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
TicketsStarted	Neue Matchmaking-Anforderungen, die seit dem letzten Bericht erstellt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
TicketsTimedOut	Matchmaking-Anforderungen, die seit dem letzten Bericht das Timeout-Limit erreicht haben. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
TimeToMatch	Für Matchmaking-Anforderungen, die vor dem letzten Bericht in einem potenziellen Match angeordnet wurden, die Dauer zwischen der Erstellung des Tickets und des potenziellen Matches. Einheiten: Sekunden Relevante CloudWatch Statistiken: Datenstichproben, Durchschnitt, Minimum, Maximum

Metrik	Description
TimeToTicketCancel	Für Matchmaking-Anforderungen, die vor dem letzten Bericht storniert wurden, die Zeit zwischen der Ticketerstellung und der Stornierung. Einheiten: Sekunden Relevante CloudWatch Statistiken: Datenstichproben, Durchschnitt, Minimum, Maximum
TimeToTicketSuccess	Für Matchmaking-Anforderungen, die vor dem letzten Bericht erfolgreich waren, die Zeit zwischen der Ticketerstellung und der erfolgreichen Match-Platzierung. Einheiten: Sekunden Relevante CloudWatch Statistiken: Datenstichproben, Durchschnitt, Minimum, Maximum

Matchmaking-Regeln

Metrik	Description
RuleEvaluationsPassed	Regelauswertungen während des Matchmaking-Prozesses, die seit dem letzten Bericht erfolgreich waren. Diese Metrik ist auf die obersten 50 Regeln begrenzt. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
RuleEvaluationsFailed	Regelauswertungen während des Matchmakings, die seit dem letzten Bericht fehlgeschlagen sind. Diese Metrik ist auf die obersten 50 Regeln begrenzt. Einheiten: Anzahl

Metrik	Description
	Relevante CloudWatch Statistiken: Summe

Amazon GameLift Servers-Metriken für FleetIQ

Der Amazon GameLift-Namespace enthält Metriken für die FleetIQ-Spielservers-Gruppe und Spielservers-Aktivität als Teil einer eigenständigen FleetIQ-Lösung für das Spiel-Hosting. Der Amazon GameLift Servers Dienst sendet Metriken an CloudWatch jede Minute. Weitere Informationen finden Sie unter [Überwachen Ihrer Auto Scaling Scaling-Gruppen und -Instances mithilfe von Amazon CloudWatch](#) im Amazon EC2 Auto Scaling-Benutzerhandbuch.

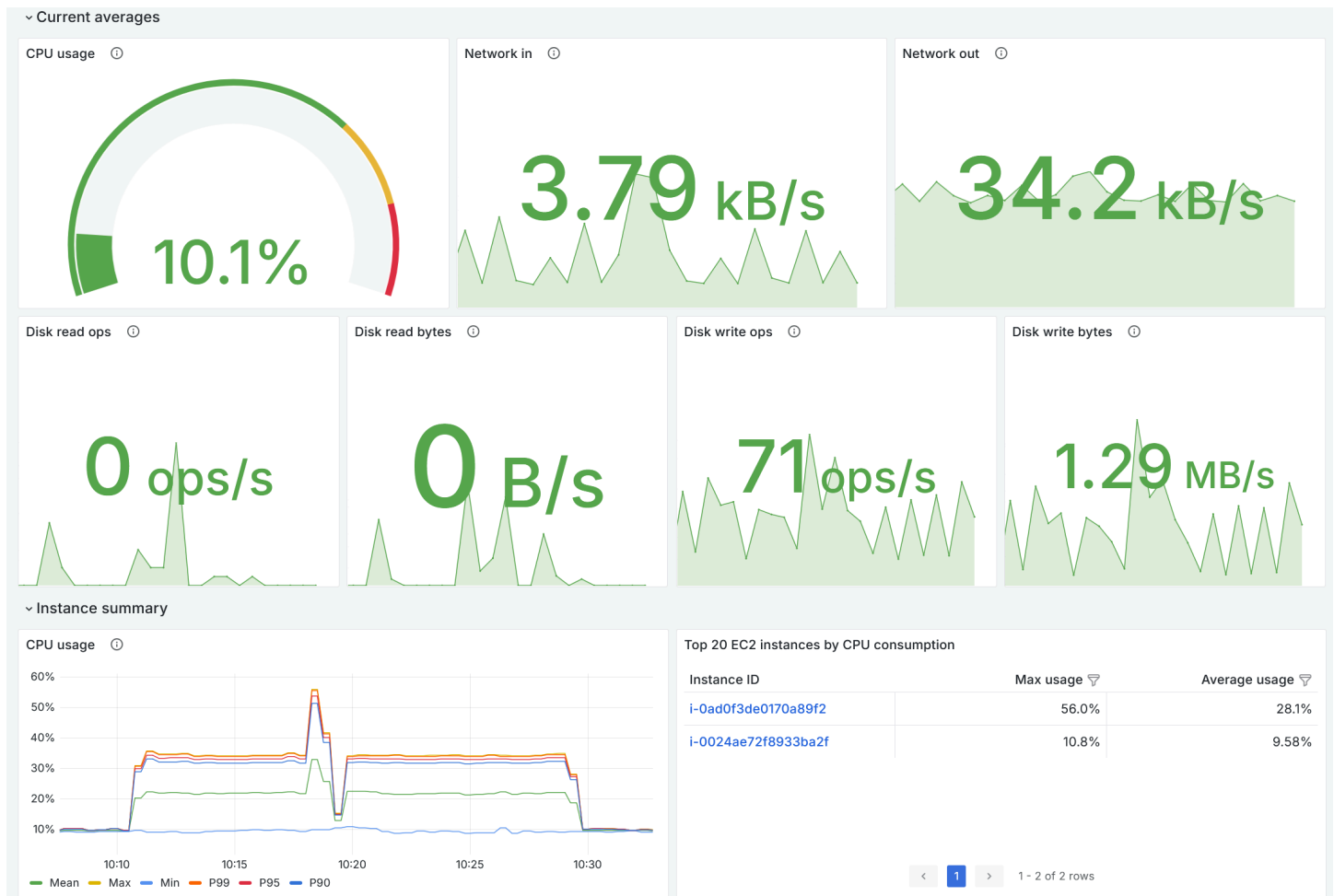
Metrik	Description
AvailableGameServers	Spiel-Server, die für Spieldurchführungen verfügbar sind und derzeit nicht mit dem Gameplay beschäftigt sind. Diese Zahl enthält Spiel-Server, die beansprucht wurden, aber immer noch den Status VERFÜGBAR haben. Einheiten: Anzahl Relevante CloudWatch Amazon-Statistiken: Summe Abmessungen: GameServerGroup
UtilizedGameServers	Spielservers, die derzeit mit Gameplay beschäftigt sind. Diese Zahl schließt Gameserver ein, die sich im Status UTILIZED befinden. Einheiten: Anzahl Relevante CloudWatch Amazon-Statistiken: Summe Abmessungen: GameServerGroup
DrainingAvailableGameServers	Spielservers auf Instances, die für die Beendigung geplant sind und die derzeit kein Gameplay unterstützen. Diese Spielservers haben die niedrigste Priorität

Metrik	Description
	, die als Reaktion auf eine neue Anspruchsanforderung beansprucht wird. Einheiten: Anzahl Relevante CloudWatch Amazon-Statistiken: Summe Abmessungen: GameServerGroup
DrainingUtilizedGameServers	Spiel-Server auf Instances, die zur Beendigung geplant sind und die derzeit das Gameplay unterstützen. Einheiten: Anzahl Relevante CloudWatch Amazon-Statistiken: Summe Abmessungen: GameServerGroup
PercentUtilizedGameServers	Teil der Spiel-Server, die derzeit Spielausführungen unterstützen. Diese Metrik gibt die Größe der Spiel-Server-Kapazität an, die derzeit verwendet wird. Sie ist nützlich, um eine Auto Scaling-Richtlinie zu steuern, die Instances dynamisch hinzufügen und entfernen kann, um den Anforderungen des Spielers gerecht zu werden. Einheiten: Prozent Relevante CloudWatch Amazon-Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: GameServerGroup

Metrik	Description
GameServerInterruptions	Spielservers auf Spot-Instances, die aufgrund der begrenzten Spot-Verfügbarkeit unterbrochen wurden. Einheiten: Anzahl Relevante CloudWatch Amazon-Statistiken: Summe Abmessungen: GameServerGroup, InstanceType
InstanceInterruptions	Spot-Instances, die aufgrund begrenzter Verfügbarkeit unterbrochen wurden. Einheiten: Anzahl Relevante CloudWatch Amazon-Statistiken: Summe Abmessungen: GameServerGroup, InstanceType

Überwachung mit Server-Telemetrie-Metriken

Amazon GameLift Servers kann so konfiguriert werden, dass Telemetriedaten für Spieleserver gesammelt und veröffentlicht werden, die auf verwalteten Amazon EC2 - und Container-Flotten laufen. Diese Metriken werden verfügbar, nachdem Sie den Telemetrie-Collector mit Ihrem Server-Build bereitgestellt haben. Das Metriksystem unterstützt alle SDKs (C++, C#, Go), alle Plugins (Unreal, Unity) und den Amazon GameLift Servers Game Server Wrapper. Metrikdaten fließen an [Amazon Managed Service for Prometheus](#) und [Amazon Managed Grafana-Dashboards](#) (zur Visualisierung empfohlen). [the section called “Überwachen Sie mit CloudWatch”](#)



Überwachen Sie die Leistung Ihrer Spieleserver mithilfe vorgefertigter Dashboards in Amazon Managed Grafana oder Amazon CloudWatch.

Vorteile von Telemetrie-Metriken

Das Telemetrie-Metriksystem bietet fünf Hauptvorteile:

- **Game-Engine-spezifische Metriken** — Game Engine-Plug-ins (Unreal, Unity) bieten eine native Integration mit Engine-spezifischen Leistungskennzahlen wie Server-Tick-Zeit, Framerate und Ressourcennutzung auf Engine-Ebene, die für die Optimierung der Spielleistung entscheidend sind.
- **Unterstützung benutzerdefinierter Metriken** — Definieren und verfolgen Sie Ihre eigenen spielspezifischen Messwerte mithilfe von Server-SDK-Funktionsaufrufen, um benutzerdefinierte Gameplay-Ereignisse, die Leistung der Geschäftslogik und anwendungsspezifische Datenpunkte zu überwachen, die für Ihr Spiel wichtig sind.

- Automatisierte Erfassung — Metriken fließen nach der Bereitstellung des Telemetrie-Collectors automatisch, ohne dass zusätzliche Instrumente erforderlich sind und direkte Integration mit [Amazon Managed Service für Prometheus](#) und Amazon erfolgt. CloudWatch
- Überwachung auf mehreren Ebenen — Metriken auf Flottenebene für Kapazität und Skalierung, Metriken auf Instanzebene für die Ressourcennutzung und Metriken für Spielsitzungen zur Leistungsverfolgung.
- Universell kompatibel — Funktioniert mit allen Amazon GameLift Servers unterstützten Entwicklungsumgebungen, ist in alle Server integriert und bietet native Unterstützung in Game SDKs Engine-Plug-ins.

Note

Telemetrie-Metriken sind für Amazon GameLift Servers verwaltete Amazon EC2 - oder Containerflotten verfügbar, auf denen Amazon Linux 2023 oder Windows ausgeführt wird.

Bevor Sie beginnen

Erforderliche Ressourcen AWS

- AWS Konto konfiguriert für Amazon GameLift Servers.
- Verwaltete Flotte läuft auf:
 - Amazon EC2 mit unterstützten Betriebssystemen, ODER
 - Container mit Amazon Linux 2023
- Entsprechende IAM-Berechtigungen

IAM-Anforderungen

Die folgenden IAM-Berechtigungen sind nur erforderlich, wenn Sie den entsprechenden Dienst nutzen möchten:

- [Amazon Managed Service für Prometheus](#) (nur erforderlich, wenn Metriken in Prometheus veröffentlicht werden)
 - `aps:RemoteWrite`-Berechtigung
 - Zugang zu Ihrem Prometheus-Workspace

- Amazon CloudWatch (nur erforderlich, wenn Metriken auf Amazon veröffentlicht werden CloudWatch)
 - `cloudwatch:PutMetricData`-Berechtigung
 - Zugriff auf Metrik-Namespaces
- [Amazon Managed Grafana](#) (nur erforderlich, wenn Grafana-Dashboards verwendet werden)
 - `grafana:Read`-Berechtigung
 - SSO-Konfiguration für den Zugriff auf das Dashboard

Implementierung

Wählen Sie Ihren Implementierungspfad basierend auf Ihrer Entwicklungsumgebung aus:

SDK-Implementierung

SDK-Typ	SDK-Einrichtung	Benutzerdefinierte Metriken	API-Referenz
Gehe zum SDK	Vollständiger Einrichtungsleitfaden	Go Metrics API	Go-Aktionen und Datentypen
C#-SDK	Vollständiger Einrichtungsleitfaden	C#-Metriken-API	C#-Aktionen und Datentypen
C++-SDK	Vollständiger Einrichtungsleitfaden	API für C++-Metriken	C++-Aktionen und Datentypen

Plugin-Implementierung

Plug-In	Plugin-Einrichtung	Benutzerdefinierte Metriken	API-Referenz
Unwirklich	Vollständiger Einrichtungsleitfaden	Unreal Metrics API	Unreale Aktionen und Datentypen
Einheit	Vollständiger Einrichtungsleitfaden	Unity Metrics API	C#-Aktionen und Datentypen

Arbeitsablauf bei der Implementierung

Jede Implementierung folgt einem zweistufigen Prozess:

1. Vollständiger Einrichtungsleitfaden (Metrics.md) — Infrastrukturbereitstellung, AWS Infrastrukturkonfiguration, Flotteneinrichtung und Grafana-Dashboard-Konfiguration.
2. API-Implementierungsleitfaden (Custom_Metrics.md) — Sprachspezifische SDK-Nutzung, Metriktypen, Erstellung benutzerdefinierter Metriken und erweiterte Konfiguration.

Verifizierung

1. Überprüfen Sie den Metrikfluss, indem Sie Ihren [Amazon Managed Service for Prometheus](#) Workspace oder Ihre CloudWatch Amazon-Konsole auf eingehende Telemetriedaten überprüfen.
2. Überprüfen Sie die Sichtbarkeit des Dashboards in [Amazon Managed Grafana](#) mithilfe der vorgefertigten Dashboards.
3. Testen Sie benutzerdefinierte Metriken, indem Sie überprüfen, ob sie in Ihren Monitoring-Dashboards angezeigt werden.

Note

Kehren Sie nach Abschluss der Implementierung zu dieser Seite zurück und gehen Sie zur Seite [Verfügbare Metriken](#).

Verfügbare Metriken

Metriken lassen sich in drei Kategorien einteilen:

- Automatisch gesammelte Metriken
- Vom SDK bereitgestellte Metriken
- Benutzerdefinierte Metriken

Automatische Erfassung von Metriken

Für diese Metriken sind keine Codeänderungen erforderlich:

Instance-Metriken

Metrik: Typ	Description	Anwendungsfall
CPU	Prozentuale Auslastung pro Instanz	Ressourcenüberwachung
Arbeitsspeicher	Physischer Speicherverbrauch und Prozentsatz	Kapazitätsplanung
Netzwerk-I/O	Gesendete/empfangene Bytes und Pakete	Zustand der Verbindung
Festplatten-IO	Lese-/Schreibvorgänge und Durchsatz	Speicherleistung

Flottenmetriken

Metrik: Typ	Description	Anwendungsfall
Aktive Instanzen	Anzahl laufender Instanzen	Skalierung der Flotte
Spielsitzungen	Aktive und verfügbare Sitzungen	Kapazitätsverwaltung
Abgestürzte Spielsitzungen	Spielsitzungen, die abgestürzt sind	Fehlerüberwachung

Vom SDK bereitgestellte Metriken

Erfordert SDK-Funktionsaufrufe in Ihrem Code:

Metriken zur Serverzeit

Metrik	Description	Implementierung
Server-Delta-Zeit	Zeitunterschied zwischen dem aktuellen Server-Tick und dem vorherigen Server-Tick. Misst die Konsistenz der Tick-Rate des Servers	Rufen Sie an <code>GetDeltaTime ()</code>

Metrik	Description	Implementierung
Server-Tick-Rate	Zeigt an, wie oft der Server pro Sekunde Updates verarbeitet	Automatisch berechnet
Server-Tick-Zeit	Die Zeit, die der Server benötigt, um einen einzelnen Tick oder ein Update zu verarbeiten	Rufen Sie an <code>GetTickCount ()</code>
Die Zeit der Serverwelt ist abgelaufen	Die Zeit, die der Server benötigt, um die Spielwelt mit jedem Tick zu aktualisieren	Rufen Sie an <code>GetWorldUpdate ()</code>

Implementierung: Für Engine-unabhängige Anwendungen SDKs (C++, C#, Go) implementieren Sie diese Metriken, indem Sie SDK-Funktionen aus Ihrem Game-Loop mit berechneten Timing-Werten aufrufen. Bei Engine-Plugins (Unreal, Unity) werden diese Metriken automatisch durch die Engine-Integration erfasst.

Netzwerkmetriken

Metrik	Description	Implementierung
Verbindungen	Die Gesamtzahl der Netzwerkverbindungen, die der Server hergestellt hat	Automatisch nach <code>InitMetrics ()</code>
Netzwerk I/O (Byte)	Die Gesamtzahl der Byte, die vom Server über das Netzwerk gesendet und empfangen wurden	Automatisch nach <code>InitMetrics ()</code>
Netzwerk I/O (Pakete)	Die Gesamtzahl der Netzwerkpakete, die vom Server gesendet und empfangen wurden	Automatisch nach <code>InitMetrics ()</code>
Verlust von Paketen	Der Prozentsatz der Netzwerkpakete, die während der Übertragung verloren gehen	Automatisch nach <code>InitMetrics ()</code>

Implementierung: Integrieren Sie SDK-Funktionsaufrufen in Ihre Netzwerkbibliothek. Das SDK bietet Anleitungen für verschiedene Netzwerkimplementierungen.

Metriken verarbeiten

Metrik	Description	Implementierung
CPU-Auslastung (%)	Der Prozentsatz der CPU-Ressourcen, die vom Spieleserverprozess genutzt werden	Automatisch nach <code>InitMetrics()</code>
Speichernutzung (Einheiten)	Die Gesamtmenge an Arbeitsspeicher, die von den Serverprozessen verbraucht wird	Automatisch nach <code>InitMetrics()</code>
Physikalische Speicherauslastung (%)	Der Prozentsatz des gesamten physischen Speichers des Servers, der derzeit genutzt wird	Automatisch nach <code>InitMetrics()</code>
Serverstatus	Zustand des Spieleservers	Automatisch nach <code>InitMetrics()</code>

Implementierung: Diese Metriken werden automatisch vom SDK für jede Spielsitzung erfasst.

Organisation des Dashboards pro Prozess

Prozessspezifische Kennzahlen sind in zwei speziellen Dashboards verfügbar:

- **Serverleistungs-Dashboard** — Enthält Server-Timings (Deltazeit, Tick-Rate, Tick-Zeit, Welt-Tick-Zeit), Netzwerkmetriken (Verbindungen, I/O Bytes/Pakete, Paketverlust), Speicherauslastung und CPU-Auslastung für einzelne Spielsitzungen.
- **Dashboard zur Instanzleistung** — Mit den Tabellen „Spielsitzungen mit den meisten Speicherbelastungen“ und „Spielsitzungen mit den meisten CPU-Ausgaben“ können Sie ermitteln, welche Prozesse am meisten zum Ressourcenverbrauch der Instanzen beitragen. Wenn Sie auf die Links zu Spielsitzungen klicken, können Sie detaillierte Messwerte genauer untersuchen.

Anwendungsfälle für Metriken pro Prozess

Die prozessualen per-game-session Metriken unterstützen die folgenden Überwachungsszenarien:

- **Umfassende Leistungsuntersuchungen** — Wenn die Leistung eines Systems aufgrund bestimmter Prozesse oder Spielsitzungen beeinträchtigt wurde, können anhand der Statistiken pro Prozess anhand der Tabellen mit den meisten CPUs- und Speicherbelastenden Spielsitzungen ermittelt werden, welcher Prozess das Problem verursacht hat.
- **Untersuchung von Spielserverabstürzen** — Wenn eine Spielsitzung abstürzt, können Sie anhand dieser Messwerte feststellen, ob der Absturz auf Speichermangel, CPU-Überlastung oder Probleme mit der Netzwerkbandbreite zurückzuführen ist.
- **Untersuchen Sie von Spielern gemeldete Probleme** — Wenn Spieler Verzögerungen oder Unterbrechungen während des Spiels melden, helfen Metriken pro Prozess dabei, Engpässe bei CPU, Arbeitsspeicher, Netzwerk, Tick-Zeit oder Welt-Update-Zeit zu identifizieren.
- **Identifizieren Sie Leistungsänderungen in verschiedenen Builds** — Mithilfe der Messwerte Ticktime, Tick-Rate und Weltupdate-Zeit können Entwickler messen, wie sich die Spieleleistung zwischen verschiedenen Server-Builds verändert.
- **Erkennen Sie Verzögerungen und Langsamkeit im Gameplay** — Die Messwerte für Ticktime, Tick-Rate und Weltupdate geben an, wie schnell der Server das Spiel aktualisiert, was sich direkt auf das Kundenerlebnis auswirkt.
- **Benchmarking** — Identifizieren Sie anhand von Faktoren wie Spielerzahl, Spielmodus und anderen Variablen, wie sich verschiedene Spielszenarien auf die Serverleistung auswirken.

Organisation des Dashboards

Metriken sind in [Amazon Managed Grafana](#) in speziellen Dashboards für verschiedene Überwachungsszenarien organisiert. Die verfügbaren Dashboards hängen von Ihrem Flottentyp ab:

EC2 Flotten-Dashboards

- **EC2 Dashboard mit Flottenübersicht** — Flottenkapazität auf hohem Niveau, Einblicke in die Skalierung, gleichzeitige Spieler (CCU), Instanzen, Spielerkapazität und abgestürzte Spielsitzungen.
- **Dashboard mit Instanzenübersicht** — Aggregierte Metriken auf Host-Ebene für alle Instances, einschließlich durchschnittlicher CPU-, Arbeitsspeicher-, Netzwerk- und Festplattenauslastung.
- **Dashboard zur Instanzleistung** — Detaillierte Metriken für einzelne Instances mit den Tabellen „Spielsitzungen mit den meisten Speicherbelastungen“ und „Spielsitzungen mit den meisten CPU-Verbrauch“ zur Identifizierung ressourcenintensiver Prozesse.
- **Dashboard zur Serverleistung (EC2)** — Timing, Netzwerkleistung, Arbeitsspeicher und CPU-Metriken für einzelne Spielsitzungen auf EC2 Instanzen.

Dashboards für Containerflotten

- Dashboard „Übersicht über die Containerflotte“ — Allgemeiner Überblick über die Auslastung der Containerflotten, einschließlich CPU-Reservierung, Speicherauslastung und Status der Containergruppen.
- Dashboard zur Container-Performance — Detaillierte Kennzahlen für einzelne Container im Rahmen bestimmter ECS-Aufgaben, einschließlich CPU-Auslastung, Speicherauslastung, Netzwerk-I/O und Speicherleistung.
- Dashboard zur Serverleistung (Container) — Timing, Netzwerkleistung, Arbeitsspeicher und CPU-Metriken für einzelne Spielsitzungen in Containern.

Ausführliche Informationen und Nutzungsanweisungen zum Dashboard finden Sie unter [the section called “Organisation und Nutzung des Dashboards”](#).

Funktionsweise

Das Telemetrie-Metriksystem folgt einem einfachen vierstufigen Datenfluss von Ihren Spieleservern zu den Visualisierungs-Dashboards.

Erfassung: Ihr Spieleserver, der in das GameLift Server-SDK oder -Plugin integriert ist, sendet automatisch Messwerte an einen lokalen Telemetrie-Collector, der auf derselben Instanz läuft. Das SDK erfasst sowohl automatische Messwerte (Serverlebenszyklus, Ressourcennutzung) als auch benutzerdefinierte Messwerte, die Sie in Ihrem Code definieren.

Verarbeitung: Der Telemetrie-Collector aggregiert Metriken von deinem Spieleserver und kombiniert sie mit Leistungsdaten auf Instanzebene (CPU, Arbeitsspeicher, Netzwerk, Festplattennutzung). Auf diese Weise erhalten Sie ein vollständiges Bild sowohl der Leistung Ihres Spiels als auch der zugrunde liegenden Infrastruktur.

Speicher: Verarbeitete Metriken werden in ein Metrik-Warehouse Ihrer Wahl exportiert — [Amazon Managed Service for Prometheus](#) für die Hochleistungsspeicherung von Zeitreihen, Amazon CloudWatch für die AWS Serviceintegration oder beides. Die gesamte Datenübertragung ist authentifiziert und verschlüsselt.

Visualisierung: [Amazon Managed Grafana](#) stellt eine Verbindung zu Ihrem Metrik-Warehouse her, um vorgefertigte Dashboards GameLift anzuzeigen. Diese Dashboards bieten Flottenübersichten, Details zur Serverleistung und Ansichten zur Container-Überwachung, mit denen Sie Ihre Spiele-Hosting-Infrastruktur überwachen und Fehler beheben können.

Note

Aus Sicherheitsgründen erfolgt die gesamte Übertragung von Metriken zwischen deinem Spieleserver und dem Telemetrie-Collector lokal auf der Instance. Nur der Collector kommuniziert mit AWS Diensten unter Verwendung der richtigen Authentifizierung.

Organisation und Nutzung des Dashboards

Sehen Sie sich Ihre Kennzahlen auf umfassenden Dashboards in [Amazon Managed Grafana](#) an. Die verfügbaren Dashboards hängen von Ihrem Flottentyp ab:

Verfügbarkeit der Dashboards nach Flottenart

Die folgende Tabelle zeigt, welche Dashboards für die einzelnen Flottenarten verfügbar sind:

Dashboard	Flottentyp	Description
EC2 Überblick über die Flotte	EC2 Flotte	Zeigt Informationen zu gleichzeitigen Spielern (CCU), Instanzen und Spielerkapazität an
Überblick über die Instanzen	EC2 Flotte	Zeigt die durchschnittliche CPU-, Arbeitsspeicher- und Netzwerkauslastung für alle Flotteninstanzen an
Instance-Leistung	EC2 Flotte	Zeigt detaillierte Metriken (CPU, Arbeitsspeicher, Festplatte, Netzwerk) für eine einzelne Instanz an
Überblick über die Containerflotte	Containerflotte	Zeigt die durchschnittliche Ressourcenauslastung aller Container in einer verwalteten Containerflotte an
Leistung von Containern	Containerflotte	Zeigt detaillierte Metriken einzelner Container innerhalb einer bestimmten ECS-Aufgabe an

Dashboard	Flottentyp	Description
Serverleistung	beides	Zeigt die Netzwerk-, Speicher- und Laufzeitleistung eines bestimmten Spielserverprozesses an (separate Versionen für Flotten EC2 und Container-Flotten)

Verwaltete EC2 Flotten:

- EC2 Die Flottenübersicht bietet umfassende Einblicke in die Flottenkapazität und Skalierung.
- Verwenden Sie die Dashboards Instances Overview und Instance Performance für die Überwachung auf Host-Ebene.
- Metriken, die über den Hostmetrics-Empfänger erfasst wurden, sorgen für Transparenz auf Systemebene.
- Konzentrieren Sie sich auf die Nutzung und Leistung der EC2 Instanzressourcen.
- Server Performance (EC2) überwacht die Anwendungsmetriken für Spielserver unabhängig von der zugrunde liegenden Infrastruktur.

Verwaltete Containerflotten:

- Verwenden Sie die Dashboards „Container Fleet Overview“ und „Container Performance“ für die Überwachung von ECS-Aufgaben und Containern.
- Mit ECS Container Receiver gesammelte Metriken sorgen für eine transparente Darstellung der Workloads in Containern.
- Konzentrieren Sie sich auf die Aggregation auf Aufgabenebene und die Isolierung von Container-Ressourcen.
- Server Performance (Container) überwacht die Anwendungsmetriken von Spielservern unabhängig von der zugrunde liegenden Infrastruktur.

EC2 Dashboard mit Flottenübersicht

Dieses Dashboard bietet einen umfassenden Überblick über die Auslastung und Kapazität Ihrer Flotte weltweit und nach Standorten. Es enthält Grafiken, die die Anzahl der Stopps, Starts und Abstürze

von Spieleservern sowie den Prozentsatz intakter Spieleserver anzeigen. Sie können nach FleetID und Standort filtern.

Kennzahlen im Überblick über die Flotte

Die folgende Tabelle zeigt die Kennzahlen, die im Flottenübersicht-Dashboard verfügbar sind:

Dashboard	Metrikname	Definition
Globale CCU und Kapazität	CCU	Zeigt die Anzahl der gleichzeitigen Benutzer auf allen Spieleservern in allen Instanzen an allen Standorten
	Globale CCU	Zeigt die Anzahl der gleichzeitigen Benutzer auf allen Spieleservern in allen Instanzen weltweit an
	Aktive Instanzen	Zeigt die Gesamtzahl der aktiven Instances in der Flotte an
	Aktive Prozesse	Zeigt die Anzahl der aktiven Spielserverprozesse an, die bereit sind, eine Spielsitzung zu hosten
	Der Spieleserver wird gestartet	Zeigt die Anzahl der Spielsitzungen an, die auf der gesamten Flotte gestartet wurden
Globale Zentraleinheit und Kapazität	Gesunde Spieleserver	Zeigt den durchschnittlichen Prozentsatz der Spieleserver in der Amazon GameLift Servers gesamten Flotte an, die als fehlerfrei gemeldet wurden
	Abgestürzte Spielsitzungen	Zeigt die Spielsitzung IDs der abgestürzten Spielsitzungen an. Klicken Sie auf den Link der Spielsitzung ARN, um zum Serverleistungs-Dashboard zu gelangen
Standort CCU	Standort Spieler	Zeigt die Anzahl der gleichzeitigen Spieler an einem Standort an, einschließlich aller

Dashboard	Metrikname	Definition
		Spieleserver und aller Instanzen an diesem Standort
	Kapazität des Standorts	Zeigt die Kapazitätsauslastung (%) an einem Standort und den Prozentsatz der an diesem Standort genutzten Spieleserver

Note

CCU-Metriken müssen in Ihrem Spieleservercode implementiert werden. Diese Messwerte werden nicht automatisch erfasst und müssen von Ihrer Anwendung implementiert und gemeldet werden.

Dashboard zur Übersicht der Instanzen

Dieses Dashboard bietet aggregierte Metriken auf Host-Ebene für alle Instances in Ihrer Flotte. Aktuelle Durchschnittswerte zeigen den Gesamtstatus der Instances. Wenn die Leistung nachlässt, überprüfen Sie die CPU-Auslastung, den Speicherverbrauch, den Netzwerk- und Festplattenverbrauch auf Engpässe. Sie können nach FleetID und Standort filtern.

Kennzahlen im Überblick über Instanzen

Die folgende Tabelle zeigt die Metriken, die im Dashboard „Instanzenübersicht“ verfügbar sind:

Dashboard	Metrikname	Definition
Zusammenfassung der Instanzen	Durchschnittliche CPU-Auslastung	In der Instanzübersicht wird die durchschnittliche Ressourcennutzung nach Instanzen nachverfolgt. Durchschnittlicher Prozentsatz der CPU-Ressourcen, die von allen Flotteninstanzen genutzt werden. Hohe Auslastungsmuster erfordern die Überprüfung der einzelnen Instanzmetriken und der Ressourcenverteilung

Dashboard	Metrikname	Definition
	CPU-Auslastung zu Spitzenzeiten	Höchste CPU-Auslastung, die in diesem Zeitraum für alle Instances in der Flotte verzeichnet wurde
	CPU-Auslastung nach Instanz	Individuelle CPU-Auslastung für jede Instanz in der Flotte, sodass Ressourcen-Hotspots identifiziert werden können
Zusammenfassung der Instanz	Durchschnittliche Speicherauslastung	Durchschnittlicher Prozentsatz des RAM, der von allen Flotteninstanzen genutzt wird. Bei hohen Auslastungsmustern müssen die Metriken einzelner Instanzen und die Verteilung der Ressourcen, z. B. der Speicherverbrauch, überprüft werden
	Speichernutzung zu Spitzenzeiten	Höchste Speicherauslastung, die in diesem Zeitraum für alle Instances in der Flotte verzeichnet wurde
	Speichernutzung nach Instance	Individuelle Speichernutzung für jede Instance in der Flotte, wodurch speicherintensive Workloads identifiziert werden können
Zusammenfassung des Netzwerks	Netzwerk I/O (Byte)	Die Netzwerkübersicht spiegelt die durchschnittliche Instanzkonnektivität wider. Das durchschnittliche Netzwerkverkehrsvolumen (gesendet und empfangen) über alle Flotteninstanzen hinweg. Überprüfen Sie bei Latenz- oder Konnektivitätsproblemen die Metriken der einzelnen Instances wie Netzwerk I/O - und Paketraten
	Netzwerk I/O (Pakete)	Durchschnittliche Netzwerkpaketrate (gesendet und empfangen) für alle Flotteninstanzen

Dashboard	Metrikname	Definition
	Netzwerknutzung nach Instanz	Individuelle Netzwerkauslastung für jede Instanz, nützlich für die Identifizierung von Netzwerkengpässen
Zusammenfassung der Festplatte	I/O Festplattenoperationen	Die Festplattenübersicht gibt die durchschnittliche Festplattenleistung der Instanzen an. Durchschnittliche read/write Festplattenoperationen für alle Flotteninstanzen. Langsame Reaktionszeiten deuten darauf hin, dass Festplattenvorgänge, I/O Wartezeiten und ausstehende Vorgänge auf der Ebene der einzelnen Instanzen untersucht werden
	I/O Festplattendurchsatz	Durchschnittlicher read/write Festplattendurchsatz für alle Flotteninstanzen
	Festplattennutzung nach Instanz	Individuelle Festplattennutzung für jede Instanz, wodurch Probleme mit der Speicherleistung identifiziert werden können

Note

Metriken auf Instance-Ebene werden über den Hostmetrics-Empfänger erfasst und bieten einen Überblick über die Infrastrukturleistung Ihrer Flotte auf Systemebene. Verwenden Sie dieses Dashboard, um allgemeine Trends zum Zustand der Flotte zu ermitteln und einzelne Fälle aufzuschlüsseln, wenn Leistungsprobleme festgestellt wurden.

Dashboard zur Instanzleistung

Dieses Dashboard bietet detaillierte Leistungskennzahlen für einzelne Instances. Aktuelle Durchschnittswerte zeigen den allgemeinen Zustand der Instance. Wenn die Leistung nachlässt, überprüfen Sie die CPU-Auslastung, den Speicherverbrauch und den Dateisystemverbrauch auf Engpässe. Es enthält Tabellen mit den meisten speicherintensiven Spielesitzungen und den meisten CPU-intensiven Spielesitzungen, anhand derer ermittelt werden kann, welche Prozesse am meisten

zum Ressourcenverbrauch der Instanzen beitragen. Wenn Sie auf die Links zu Spielsitzungen klicken, können Sie detaillierte Messwerte genauer untersuchen. Du kannst nach einer bestimmten Instanz-ID filtern.

Kennzahlen zur Instanz-Leistung

Die folgende Tabelle zeigt die Metriken, die im Dashboard zur Instanzleistung verfügbar sind:

Dashboard	Metrikname	Definition
Lade-Zusammenfassung	Spielsitzungen, die am meisten Arbeitsspeicher beanspruchen	In der Ladeübersicht wird die Ressourcennutzung durch die Spieleserver nachverfolgt. Rangliste der Spielsitzungen, die in dieser Instanz die meisten Speicherressourcen verbrauchen. Bei hohen Auslastungsmustern müssen die einzelnen Sitzungsmetriken und die Ressourcenverteilung überprüft werden. Ein Klick auf die Links zur Spielsitzung ermöglicht eine genauere Untersuchung detaillierter Messwerte
	Die meisten CPU-intensiven Spielsitzungen	Rangliste der Spielsitzungen, die in dieser Instanz die meisten CPU-Ressourcen verbrauchen. Bei hohen Auslastungsmustern müssen die einzelnen Sitzungsmetriken und die Ressourcenverteilung wie CPU und Arbeitsspeicher überprüft werden. Ein Klick auf die Links zur Spielsitzung ermöglicht eine genauere Untersuchung detaillierter Messwerte
	CPU-Auslastung pro Spielsitzung	Aufschlüsselung der individuellen CPU-Auslastung mit Angabe des Ressourcenverbrauchs nach jeder aktiven Spielsitzung
	Speichernutzung pro Spielsitzung	Aufschlüsselung der individuellen Speichernutzung mit Angabe des Ressourcenverbrauchs nach jeder aktiven Spielsitzung

Dashboard	Metrikname	Definition
Aktuelle Durchschnittswerte	CPU-Auslastung der Instanz	Gesamt-CPU-Auslastung für die ausgewählte Instance im Laufe der Zeit
	Speichernutzung der Instanz	Gesamtspeichernutzung für die ausgewählte Instance im Laufe der Zeit
	Nutzung des Instanz-Dateisystems	Auslastung des Dateisystems für die ausgewählte Instanz, nützlich bei der Identifizierung von Speicherkapazitätsproblemen
Netzwerk	Netzwerk-I/O der Instanz	Das Netzwerk spiegelt die Instanzkonnektivität wider. Volumen des Netzwerkverkehrs und Paketraten für die ausgewählte Instance. Untersuchen Sie bei Latenz- oder Konnektivitätsproblemen die Netzwerk-I/O, die Paketraten und die Anzahl der Fehler
	Netzwerkverbindungen	Anzahl der aktiven Netzwerkverbindungen auf der ausgewählten Instanz
	Zählt Netzwerkfehler	Netzwerkfehlerstatistiken zur Identifizierung von Verbindungsproblemen
Festplatte	I/O Festplattenoperationen	Die Festplatte zeigt die Festplattenleistung an. read/write Festplattenaktivität und Auslastung für die ausgewählte Instanz. Langsame Reaktionszeiten deuten darauf hin, dass Festplattenvorgänge, I/O Wartezeiten und ausstehende Vorgänge untersucht werden
	I/O Wartezeiten auf Festplatten	Wartezeit auf den Abschluss der Festplattenoperationen
	Ausstehende Festplattenoperationen	Anzahl der Festplattenoperationen, die auf ihre Verarbeitung warten

Note

Die Tabellen mit den meisten Arbeitsspeicher- und CPU-intensiven Spielsitzungen sind wichtig, um Leistungsengpässe und ressourcenintensive Prozesse zu identifizieren, die sich auf die Gesamtleistung der Instance auswirken können. Diese Ranglisten ermöglichen eine schnelle Identifizierung problematischer Spielsitzungen zur weiteren Untersuchung.

Dashboard zur Übersicht über die Containerflotte

Dieses Dashboard bietet einen umfassenden Überblick über die Ressourcennutzung und Kapazität Ihrer Containerflotte. Es zeigt die durchschnittliche Ressourcennutzung aller Container in einer verwalteten Containerflotte an, einschließlich CPU-Reservierung, Speicherauslastung und Status der Containergruppen. Sie können nach FleetID und Standort filtern.

Kennzahlen zur Übersicht über die Containerflotte

Die folgende Tabelle zeigt die Kennzahlen, die im Dashboard „Container Fleet Overview“ verfügbar sind:

Dashboard	Metrikname	Definition
Status der Container-Gruppe	Aktive Spielserver-Containergruppen	Container-Gruppen, die derzeit aktiv und bereit sind, Spielsitzungen abzuhalten
	Containergruppen auf Spieleservern im Leerlauf	Container-Gruppen, die aktiv sind, aber derzeit keine Spielsitzungen veranstalten
	Ausstehende Gameserver-Containergruppen	Container-Gruppen, die gerade gestartet werden
	Gameserver-Containergruppen werden beendet	Container-Gruppen, die gerade heruntergefahren werden
Nutzung der Ressourcen	CPU-Auslastung von Containern	Durchschnittliche CPU-Auslastung aller Container in der Flotte

Dashboard	Metrikname	Definition
	Auslastung des Container-Speichers	Durchschnittliche Speicherauslastung für alle Container in der Flotte
	CPU-Reservierung für Container	Prozentsatz der CPU-Ressourcen, die in der gesamten Flotte von Containern reserviert wurden
Netzwerkaktivität	Container-Netzwerk eingehend	Durchschnittliche Anzahl an Netzwerk-Bytes, die von Containern innerhalb der Flotte empfangen wurden
	Container-Netzwerk ausgefallen	Durchschnittliche Anzahl an Netzwerk-Bytes, die von Containern innerhalb der Flotte gesendet werden

Note

Metriken zur Containerflotte werden über ECS Container Receiver erfasst und bieten einen Überblick über die Workloads in Containern, wobei der Schwerpunkt auf der Aggregation auf Aufgabenebene und der Isolierung von Container-Ressourcen liegt.

Dashboard zur Container-Leistung

Dieses Dashboard bietet detaillierte Leistungskennzahlen für einzelne Container innerhalb bestimmter ECS-Aufgaben. Es zeigt detaillierte Metriken einzelner Container an, darunter CPU-Auslastung, Speicherauslastung, Netzwerk-I/O und Speicherleistung. Sie können nach einer bestimmten Container-ID oder ECS-Task filtern.

Kennzahlen zur Container-Leistung

Die folgende Tabelle zeigt die Metriken, die im Container Performance-Dashboard verfügbar sind:

Dashboard	Metrikname	Definition
Leistung der Ressourcen	CPU-Auslastung von Containern	CPU-Auslastung für den ausgewählten Container im Laufe der Zeit
	Speicherauslastung des Containers	Speicherauslastung für den ausgewählten Container im Laufe der Zeit
	Reservierung des Container-Speichers	Prozentsatz der vom ausgewählten Container reservierten Speicherressourcen
Netzwerkleistung	Container-Netzwerk-I/O	Volumen des Netzwerkverkehrs (ein-/ausgehende Byte) für den ausgewählten Container
	Netzwerkverbindungen	Anzahl der aktiven Netzwerkverbindungen für den ausgewählten Container
Leistung des Speichers	Lesen/Schreiben im Containerspeicher	Lese- und Schreibaktivität für den ausgewählten Container speichern
	I/O Vorgänge bei der Lagerung von Containern	Anzahl der I/O Speichervorgänge, die vom ausgewählten Container ausgeführt wurden

Note

Container-Leistungskennzahlen bieten einen detaillierten Einblick in den Ressourcenverbrauch einzelner Container und die Leistungsmerkmale innerhalb von ECS-Aufgaben.

Dashboard zur Serverleistung

Das Serverleistungs-Dashboard zeigt Messwerte zu Serverzeiten, Netzwerkaktivität, Arbeitsspeicher und CPU-Auslastung für einzelne Spielsitzungen. Sie können nach der Game Session ID filtern und Metriken direkt nach Amazon CloudWatch oder [Amazon Managed Grafana](#) exportieren.

Metriken zur Serverleistung

Die folgende Tabelle zeigt die Messwerte, die im Serverleistungs-Dashboard verfügbar sind:

Dashboard	Metrikname	Definition
Server-Timings	Server-Delta-Zeit	Zeitunterschied zwischen dem aktuellen Server-Tick und dem vorherigen Server-Tick. Dies ist ein Maß für die Konsistenz der Tick-Rate des Servers
	Server-Tick-Rate	Zeigt an, wie oft der Server pro Sekunde Updates verarbeitet
	Server-Tick-Zeit	Die Zeit, die der Server benötigt, um einen einzelnen Tick oder ein Update zu verarbeiten
	Zeit für das weltweite Ticken auf dem Server	Die Zeit, die der Server benötigt, um die Spielwelt mit jedem Tick zu aktualisieren
Netzwerk	Verbindungen	Die Gesamtzahl der Netzwerkverbindungen, die der Server hergestellt hat
	Netzwerk I/O (Byte)	Die Gesamtzahl der Byte, die vom Server über das Netzwerk gesendet und empfangen wurden
	Netzwerk I/O (Pakete)	Die Gesamtzahl der Netzwerkpakete, die vom Server gesendet und empfangen wurden
	Paketverlust	Der Prozentsatz der Netzwerkpakete, die während der Übertragung verloren gehen
Arbeitsspeicher	Speichernutzung (Einheiten)	Die Gesamtmenge an Arbeitsspeicher, die von den Serverprozessen verbraucht wird

Dashboard	Metrikname	Definition
	Physikalische Speicherauslastung (%)	Der Prozentsatz des gesamten physischen Speichers des Servers, der derzeit genutzt wird
CPU-Verwendung	CPU-Auslastung (%)	Der Prozentsatz der CPU-Ressourcen, die vom Spieleserverprozess genutzt werden

Allgemeine Überwachungsszenarien

Tauchen Sie tief in die Leistungsuntersuchung ein

Szenario: A weist host/instance aufgrund bestimmter Prozesse oder Spielsitzungen eine Leistungseinbuße auf

Schritte der Untersuchung:

- Rufen Sie das Dashboard zur Instanzleistung auf.
- Sehen Sie sich die Tabelle „Spielsitzungen mit den meisten Speicherbelastungen“ an, um herauszufinden, welche Prozesse am meisten zum Speicherverbrauch der Instances beitragen.
- In der Tabelle mit den meisten CPU-intensiven Spielsitzungen finden Sie heraus, welche Prozesse am meisten zur CPU-Auslastung der Instanzen beitragen.
- Klicken Sie auf die Links zu Spielsitzungen, um eine genauere Untersuchung detaillierter Messwerte zu erhalten.
- Analysieren Sie die Serverzeiten (Server Delta Time, Server Tick Rate, Server Tick Time, Server World Tick Time), um Leistungsengpässe zu identifizieren.

Untersuchung von Spieleserverabstürzen

Szenario: Eine Spielsitzung ist abgestürzt und Sie müssen die Ursache ermitteln

Schritte der Untersuchung:

- Greifen Sie auf das Serverleistungs-Dashboard für die abgestürzte Spielsitzung zu.
- Überprüfe die Speichernutzung (Einheiten) und die physische Speicherauslastung (%), um festzustellen, ob der Absturz auf zu wenig Arbeitsspeicher zurückzuführen ist.

- Überprüfen Sie die CPU-Auslastung (%), um festzustellen, ob die CPU-Überlastung den Absturz verursacht hat.
- Analysieren Sie Netzwerk I/O (Byte) und Netzwerk I/O (Pakete), um festzustellen, ob Probleme mit der Netzwerkbandbreite zum Absturz beigetragen haben.
- Untersuchen Sie den Prozentsatz des Paketverlusts, um Netzwerkprobleme zu identifizieren.

Untersuchen Sie die vom Spieler gemeldeten Probleme

Szenario: Spieler melden Verzögerungen oder Unterbrechungen während des Spiels

Schritte der Untersuchung:

- Greifen Sie auf das Serverleistungs-Dashboard für die betroffene Spielsitzung zu.
- Überprüfe Server Tick Time und Server World Tick Time, um Verzögerungen bei Spielupdates zu erkennen.
- Überprüfe die Server-Tick-Rate, um eine gleichbleibende Häufigkeit der Server-Updates sicherzustellen.
- Analysieren Sie die CPU-Auslastung (%), um Verarbeitungsengpässe zu identifizieren.
- Überprüfen Sie die Kennzahlen zur Speicherauslastung, um speicherbezogene Leistungsprobleme zu identifizieren.
- Überprüfen Sie I/O Netzwerkmetriken und Paketverlust, um Netzwerkengpässe zu identifizieren.

Identifizieren Sie Leistungsänderungen in verschiedenen Spielserver-Builds

Szenario: Sie möchten messen, wie sich die Spielleistung zwischen verschiedenen Server-Builds verändert

Schritte der Untersuchung:

- Vergleichen Sie die Server Tick Time-Metriken zwischen verschiedenen Builds, um Änderungen der Verarbeitungseffizienz zu messen.
- Analysieren Sie die Konsistenz der Server-Tick-Rate zwischen Builds, um Leistungseinbußen zu identifizieren.
- Überprüfen Sie die Server World Tick Time, um die Leistungsänderungen bei Spielwelt-Updates zu messen.

- Vergleichen Sie die Muster der Speichernutzung zwischen Builds, um Verbesserungen oder Regressionen bei der Speicheroptimierung zu ermitteln.
- Überwachen Sie Trends bei der CPU-Auslastung, um Veränderungen der Recheneffizienz zu beurteilen.

Erkenne Verzögerungen und Langsamkeit im Gameplay

Szenario: Sie müssen die Reaktionsfähigkeit des Servers und die Geschwindigkeit der Spielupdates überwachen

Schritte der Untersuchung:

- Überwachen Sie die Server-Tick-Zeit, um zu messen, wie schnell der Server jeden Aktualisierungszyklus verarbeitet.
- Verfolge die Server-Tick-Rate, um konsistente Aktualisierungen des Spielstatus pro Sekunde sicherzustellen.
- Analysieren Sie die Server World Tick Time, um die Aktualisierungsgeschwindigkeit der Spielwelt zu messen, was sich direkt auf das Kundenerlebnis auswirkt.
- Richten Sie Warnmeldungen für Schwankungen der Server-Deltazeit ein, um eine inkonsistente Serverleistung zu erkennen.

Benchmarking verschiedener Spielszenarien

Szenario: Sie möchten herausfinden, wie sich verschiedene Spielszenarien auf die Serverleistung auswirken

Schritte der Untersuchung:

- Vergleichen Sie die Serverleistungskennzahlen verschiedener Spieler, um die Auswirkungen der Skalierung zu verstehen.
- Analysieren Sie Leistungsunterschiede zwischen den Spielmodi mithilfe von Server-Tick-Time- und CPU-Nutzungsmetriken.
- Überwachen Sie die Muster der Speichernutzung in verschiedenen Spielszenarien, um ressourcenintensive Funktionen zu identifizieren.
- Verfolge I/O Netzwerkmetriken, um die Bandbreitenanforderungen für verschiedene Spielszenarien zu verstehen.

- Ermitteln Sie anhand des Dashboards zur Instanzleistung, welche Spielszenarien zu den ressourcenintensivsten Spielsitzungen führen.

Reaktion bei hoher Ressourcenauslastung

Szenario: Ungewöhnliche Ressourcenspitzen (CPU > 85%, Arbeitsspeicher > 90%)

Schritte der Untersuchung:

Identifizieren Sie die betroffenen Ressourcen

- Verwenden Sie die DescribeGameSessionDetails API.
- Filtern Sie bei Bedarf nach Status.
- Dokumentieren Sie die betroffenen Instanzen.

Analysieren Sie die Ressourcennutzung

- Überprüfen Sie das Dashboard mit der Instanzübersicht.
- Vergleichen Sie die Auslastung zwischen den einzelnen Flotten.
- Prüfen Sie historische Muster.

Überwachen Sie die Auswirkungen auf den Spielserver

- Überprüfe die Metriken zur Serverleistung.
- Überprüfen Sie die Tick-Zeiten und den Paketverlust.
- Überwachen Sie Speicherlecks.

Lösungsschritte

- Laden Sie Sitzungsprotokolle herunter.
- Beheben Sie Probleme beim Erstellen.
- Überwachen Sie die Verbesserungen.

Absturzanalyse des Spieleservers

Szenario: Mehrere Spielsitzungen mit Fehlerstatus in der gesamten Flotte

Schritte der Untersuchung:

Erste Bewertung

- Rufen Sie das Dashboard „Flottenübersicht“ auf.
- Überprüfen Sie die Tabelle der abgestürzten Sitzungen.
- Beachten Sie Muster in Bezug auf Zeit und Ort.

Leistungsanalyse

- Überprüfen Sie die Server-Timing-Metriken.
- Überprüfen Sie die Ressourcennutzung.
- Überwachen Sie die Netzwerkleistung.

Überprüfung der Infrastruktur

- Überprüfen Sie die Flottenkapazität.
- Überprüfen Sie den Zustand der Instanz.
- Überprüfen Sie die Skalierungsrichtlinien.

Lösungspfad

- Analysieren Sie Serverprotokolle.
- Überprüfen Sie die Code-Optimierung.
- Implementieren Sie Korrekturen.

Optimierung der Flottenkapazität

Szenario: Spielstart oder Benchmark-Studie

Schritte der Analyse:

Ressourcenauslastung

- Nach Standort filtern.
- Überprüfen Sie die P50/P95/P99-Metriken.
- Analysieren Sie Nutzungsmuster.

Analyse des Instanztyps

- Vergleichen Sie die Leistung nach Typ.
- Identifizieren Sie Skalierungskandidaten.
- Nutzungsmuster von Dokumenten.

Optimierungsmaßnahmen

- Passen Sie die Skalierungsrichtlinien an.
- Ändern Sie die Instanztypen.
- Aktualisieren Sie die Flottenkonfiguration.

Anleitung zur Fehlerbehebung

Allgemeine Probleme und Lösungsschritte

Fehlende oder unvollständige Metriken

Symptome

- In den Dashboards werden keine Metriken angezeigt.
- Teilweise Erfassung von Metriken.
- Verzögerte Metrikaktualisierungen.

Lösungsschritte

A. Überprüfen Sie den Collector-Status

Überprüfen Sie den Systemd-Dienst:

```
sudo systemctl status gamelift-telemetry-collector
```

Collector-Protokolle überprüfen:

```
sudo journalctl -u gamelift-telemetry-collector
```

- Bestätigen Sie die Collector-Konfiguration.

B. Überprüfung der IAM-Berechtigungen

- Überprüfen Sie die Berechtigungen für die Instanzrolle.
- Überprüfen Sie die erforderlichen Richtlinien:
 - `aps:RemoteWrite`
 - `cloudwatch:PutMetricData`
- Überprüfen Sie die Vertrauensbeziehungen der Rollen.

C. Netzwerkkonnektivität

- Überprüfen Sie den Endpunktzugriff.
- Überprüfen Sie die Sicherheitsgruppenregeln.
- Überprüfen Sie das Netzwerk ACLs.

Authentifizierungsfehler

Symptome

- SigV4-Authentifizierungsfehler.
- Zugriff auf Nachrichten verweigert.
- Probleme bei der Aktualisierung der Anmeldeinformationen.

Lösungsschritte

A. SigV4-Authentifizierung

- Überprüfen Sie die temporären Anmeldeinformationen.
- Überprüfen Sie die Rotation der Anmeldeinformationen.
- Überprüfen Sie das Instanzprofil.

B. AMP-Zugriff

- Überprüfen Sie die Workspace-Konfiguration.
- Überprüfen Sie die Remote-Schreib-URL.
- Überprüfen Sie die IAM-Rollenbindungen.

Probleme mit dem Dashboard

Symptome

- Leere Dashboards.
- Fehlende Datenpunkte.
- Fehler bei der Authentifizierung.

Lösungsschritte

A. Konfiguration der Datenquelle

- Überprüfen Sie die Prometheus-Verbindung.
- Prüfen Sie die CloudWatch Amazon-Integration.
- Testen Sie die Datenquellenberechtigungen.

B. Grafana-Zugang

- Bestätigen Sie die SSO-Konfiguration.
- Überprüfen Sie bei Bedarf die 2FA-Einrichtung.
- Überprüfen Sie die Benutzerberechtigungen.

Windows-spezifische Probleme

Symptome

- Fehler beim Starten des Dienstes.
- Lücken bei der Erfassung von Metriken.
- Fehler bei der Genehmigung.

Lösungsschritte

- Überprüfen Sie den Windows-Dienststatus.
- Überprüfen Sie die Windows-Ereignisprotokolle.
- Überprüfen Sie die Collector-Konfiguration.
- Überprüfen Sie Windows-spezifische Pfade.

Amazon GameLift Servers API-Aufrufe protokollieren mit AWS CloudTrail

Amazon GameLift Servers ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt wurden Amazon GameLift Servers. CloudTrail erfasst alle API-Aufrufe Amazon GameLift Servers als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Amazon GameLift Servers-Konsole und Code-Aufrufe der Amazon GameLift Servers-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Amazon GameLift Servers. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage ermitteln CloudTrail, an die die Anfrage gestellt wurde Amazon GameLift Servers, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details.

Weitere Informationen CloudTrail dazu finden Sie im [AWS CloudTrail Benutzerhandbuch](#).

Amazon GameLift Servers Informationen in CloudTrail

CloudTrail ist auf Ihrem aktiviert AWS-Konto , wenn Sie das Konto erstellen. Wenn eine Aktivität in stattfindet Amazon GameLift Servers, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem anzeigen, suchen und herunterladen AWS-Konto. Weitere Informationen finden Sie unter [Ereignisse mit dem CloudTrail Ereignisverlauf anzeigen](#).

Zur kontinuierlichen Aufzeichnung von Ereignissen in Ihrem AWS-Konto, einschließlich Ereignissen für Amazon GameLift Servers, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- [Übersicht zum Erstellen eines Trails](#)
- [CloudTrail unterstützte Dienste und Integrationen](#)

- [Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail](#)
- [Empfangen von CloudTrail Protokolldateien aus mehreren Regionen](#) und [Empfangen von CloudTrail Protokolldateien von mehreren Konten](#)

Alle Amazon GameLift Servers Aktionen werden von der [Amazon GameLift ServersAPI-Referenz](#) protokolliert CloudTrail und sind in dieser dokumentiert. Beispielsweise generieren Aufrufe CreatePlayerSession und UpdateGameSession Aktionen Einträge in den CloudTrail Protokolldateien. CreateGameSession

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 Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter [CloudTrail -Element userIdentity](#).

Grundlagen zu Amazon GameLift Servers-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die DescribeFleetAttributes Aktionen CreateFleet und demonstriert.

```
{
  "Records": [
    {
      "eventVersion": "1.04",
      "userIdentity": {
        "type": "IAMUser",
```

```

    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/myUserName",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "myUserName"
  },
  "eventTime": "2015-12-29T23:40:15Z",
  "eventSource": "gamelift.amazonaws.com",
  "eventName": "CreateFleet",
  "awsRegion": "us-west-2",
  "sourceIpAddress": "192.0.2.0",
  "userAgent": "[]",
  "requestParameters": {
    "buildId": "build-92b6e8af-37a2-4c10-93bd-4698ea23de8d",
    "eC2InboundPermissions": [
      {
        "ipRange": "10.24.34.0/23",
        "fromPort": 1935,
        "protocol": "TCP",
        "toPort": 1935
      }
    ],
    "logPaths": [
      "C:\\game\\serverErr.log",
      "C:\\game\\serverOut.log"
    ],
    "eC2InstanceType": "c5.large",
    "serverLaunchPath": "C:\\game\\MyServer.exe",
    "description": "Test fleet",
    "serverLaunchParameters": "-paramX=baz",
    "name": "My_Test_Server_Fleet"
  },
  "responseElements": {
    "fleetAttributes": {
      "fleetId": "fleet-0bb84136-4f69-4bb2-bfec-a9b9a7c3d52e",
      "serverLaunchPath": "C:\\game\\MyServer.exe",
      "status": "NEW",
      "logPaths": [
        "C:\\game\\serverErr.log",
        "C:\\game\\serverOut.log"
      ],
      "description": "Test fleet",
      "serverLaunchParameters": "-paramX=baz",
      "creationTime": "Dec 29, 2015 11:40:14 PM",

```

```

        "name": "My_Test_Server_Fleet",
        "buildId": "build-92b6e8af-37a2-4c10-93bd-4698ea23de8d"
    },
    {
        "requestID": "824a2a4b-ae85-11e5-a8d6-61d5cafb25f2",
        "eventID": "c8fbea01-fbf9-4c4e-a0fe-ad7dc205ce11",
        "eventType": "AwsApiCall",
        "recipientAccountId": "111122223333"
    },
    {
        "eventVersion": "1.04",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "AIDACKCEVSQ6C2EXAMPLE",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
        "eventTime": "2015-12-29T23:40:15Z",
        "eventSource": "gamelift.amazonaws.com",
        "eventName": "DescribeFleetAttributes",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "[]",
        "requestParameters": {
            "fleetIds": [
                "fleet-0bb84136-4f69-4bb2-bfec-a9b9a7c3d52e"
            ]
        },
        "responseElements": null,
        "requestID": "82e7f0ec-ae85-11e5-a8d6-61d5cafb25f2",
        "eventID": "11daabcb-0094-49f2-8b3d-3a63c8bad86f",
        "eventType": "AwsApiCall",
        "recipientAccountId": "111122223333"
    },
]
}

```

Servermeldungen protokollieren Amazon GameLift Servers

Sie können benutzerdefinierte Servermeldungen von Ihren Amazon GameLift Servers Servern in Protokolldateien erfassen. Die Art und Weise, wie Sie die Protokollierung konfigurieren, hängt davon

ab Amazon GameLift Servers Realtime, ob Sie benutzerdefinierte Server oder verwaltete EC2 Flotten oder Containerflotten verwenden (siehe die entsprechenden Unterabschnitte in diesem Kapitel).

Themen

- [Unterschiede protokollieren: Verwaltete Flotten EC2 und Containerflotten](#)
- [Servermeldungen protokollieren \(benutzerdefinierte Server\)](#)

Unterschiede protokollieren: Verwaltete Flotten EC2 und Containerflotten

Das Protokollierungsverhalten unterscheidet sich zwischen verwalteten EC2 Flotten und Containerflotten erheblich. Das Verständnis dieser Unterschiede ist entscheidend, wenn Sie von EC2 zu Containern migrieren oder den richtigen Flottentyp für Ihre Protokollierungsanforderungen auswählen.

Überblick über die wichtigsten Unterschiede

Feature	Verwaltete EC2 Flotten	Containerflotten
Aufbewahrung von Protokollen	Automatischer Upload auf Amazon S3 nach Ende der Spielsitzung	Amazon CloudWatch Logs oder eine benutzerdefinierte Protokollierungslösung muss konfiguriert werden
GetGameSessionLogUrl API	Verfügbar — gibt die Amazon S3 S3-URL zum Herunterladen des Protokolls zurück	Nicht verfügbar — Protokolle werden nicht automatisch in Amazon S3 gespeichert
Aufbewahrung von Protokollen	14 Tage in Amazon S3 (automatisch)	Hängt von Ihrer Protokollierungskonfiguration ab
Überwachung in Echtzeit	Eingeschränkt — Protokolle sind erst nach Ende der Spielsitzung verfügbar	Verfügbar mit Amazon CloudWatch Logs-Integration
Komplexität der Einrichtung	Automatisch — keine zusätzliche Konfiguration erforderlich	Erfordert eine explizite Protokollierungskonfiguration

Verwaltete EC2 Flottenprotokollierung

Amazon GameLift Servers bietet automatische Protokollverwaltung für verwaltete EC2 Flotten:

- Automatischer Upload: Serverprotokolle werden automatisch auf Amazon S3 hochgeladen, wenn eine Spielsitzung endet
- GetGameSessionLogUrl API: Verwenden Sie diese API, um eine vorsignierte URL zum Herunterladen von Protokollen von Amazon S3 abzurufen
- Aufbewahrung von 14 Tagen: Protokolle werden 14 Tage lang in Amazon S3 aufbewahrt, bevor sie automatisch gelöscht werden
- Größenbeschränkungen: Für Protokolldateien gelten Größenbeschränkungen pro Spielsitzung (siehe [Amazon GameLift Servers Endpunkte und Kontingente](#))

Weitere Informationen finden Sie unter [the section called “Protokollierung für benutzerdefinierte Server”](#) und in der [GetGameSessionLogUrl API-Referenz](#).

Protokollierung von Containerflotten

Bei Containerflotten müssen Sie die Protokollierung explizit konfigurieren:

- Kein automatischer Amazon S3-Upload: Protokolle werden nicht automatisch auf Amazon S3 hochgeladen
- GetGameSessionLogUrl nicht verfügbar: Diese API funktioniert nicht mit Containerflotten
- Amazon CloudWatch Logs-Integration: Konfigurieren Sie Ihren Container so, dass er Protokolle zur zentralen CloudWatch Protokollierung an Amazon Logs sendet
- Maßgeschneiderte Logging-Lösungen: Implementieren Sie Ihre eigene Logging-Infrastruktur mithilfe von Log-Treibern oder Sidecar-Containern
- Echtzeitzugriff: Bei richtiger Konfiguration kann während der Spielsitzungen in Echtzeit auf Protokolle zugegriffen werden

Ausführliche Optionen für die Protokollierung von Containern finden Sie unter [Funktionsweise von Containerflotten](#).

Überlegungen zur Migration

Beachten Sie bei der Migration von verwalteten Flotten EC2 zu Container-Flotten die folgenden Änderungen an der Protokollierung:

- Protokollabrufcode aktualisieren: Ersetzen Sie `GetGameSessionLogUrl` API-Aufrufe durch Amazon CloudWatch Logs-Abfragen oder Ihre benutzerdefinierte Protokollierungslösung
- Protokollaufbewahrung konfigurieren: Richten Sie entsprechende Aufbewahrungsrichtlinien in Amazon CloudWatch Logs oder Ihrem Protokollierungssystem ein
- Implementieren Sie eine Echtzeitüberwachung: Nutzen Sie den Protokollzugriff in Echtzeit für eine bessere Beobachtbarkeit
- Überprüfen Sie das Protokollvolumen und die Kosten: Die Preise von Amazon CloudWatch Logs unterscheiden sich vom unbegrenzten Amazon S3 S3-Speicher in verwalteten EC2 Flotten

Empfohlene Einrichtung der Container-Protokollierung

Für Containerflotten empfehlen wir:

1. Amazon CloudWatch Logs-Integration: Konfigurieren Sie Ihre Container-Definition für die Verwendung des `awslogs` Protokolltreibers
2. Strukturierte Protokollierung: Verwenden Sie strukturierte Protokollformate (JSON) für eine bessere Durchsuchbarkeit und Analyse
3. Protokollebenen: Implementieren Sie geeignete Protokollebenen, um die Ausführlichkeit und die Kosten zu kontrollieren
4. Aufbewahrungsrichtlinien: Legen Sie Aufbewahrungsfristen auf der Grundlage Ihrer Compliance- und Debugging-Anforderungen fest

Servermeldungen protokollieren (benutzerdefinierte Server)

Sie können benutzerdefinierte Servermeldungen von Ihren Amazon GameLift Servers benutzerdefinierten Servern in Protokolldateien erfassen.

Important

Die Größe einer Protokolldatei pro Spielsitzung ist begrenzt (siehe [Amazon GameLift Servers Endpunkte und Kontingente](#) im Allgemeine AWS-Referenz). Wenn eine Spielsitzung endet, werden die Serverprotokolle auf Amazon Simple Storage Service (Amazon S3) Amazon GameLift Servers hochgeladen. Amazon GameLift Servers lädt keine Protokolle hoch, die das Limit überschreiten. Protokolle können sehr schnell wachsen und die

Größenbeschränkung überschreiten. Sie sollten Ihre Protokolle überwachen und die Protokollausgabe auf die erforderlichen Meldungen beschränken.

Konfiguration der Protokollierung für benutzerdefinierte Server

Bei Amazon GameLift Servers benutzerdefinierten Servern schreiben Sie Ihren eigenen Code für die Protokollierung, den Sie als Teil Ihrer Serverprozesskonfiguration konfigurieren. Amazon GameLift Servers verwendet Ihre Protokollierungskonfiguration, um die Dateien zu identifizieren, die am Ende jeder Spielsitzung auf S3 hochgeladen werden müssen.

Die folgenden Anweisungen zeigen anhand von vereinfachten Codebeispielen, wie die Protokollierung konfiguriert wird:

C++

So konfigurieren Sie die Protokollierung (C++)

1. Erstellen Sie einen Vektor von Zeichenketten, die Verzeichnispfade zu den Protokolldateien des Spieleservers sind.

```
std::string serverLog("serverOut.log");           // Example server log file
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
```

2. Geben Sie Ihren Vektor als den [LogParameters](#) Ihres [ProcessParameters](#) Objekts an.

```
Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
        listenPort,
        Aws::GameLift::Server::LogParameters(logPaths));
```

3. Geben Sie das [ProcessParameters](#) Objekt an, wenn Sie [ProcessReady\(\)](#) aufrufen.

```
Aws::GameLift::GenericOutcome outcome =
    Aws::GameLift::Server::ProcessReady(processReadyParameter);
```

Ein vollständigeres Beispiel finden Sie unter [ProcessReady\(\)](#).

C#

So konfigurieren Sie die Protokollierung (C#)

1. Erstellen Sie eine Liste von Zeichenfolgen, die Verzeichnispfade zu den Protokolldateien des Spieleservers sind.

```
List<string> logPaths = new List<string>();  
logPaths.Add("C:\\\\game\\\\serverOut.txt");    // Example of a log file that the  
game server writes
```

2. Geben Sie Ihre Liste als die [LogParameters](#) Ihres [ProcessParameters](#) Objekts an.

```
var processReadyParameter = new ProcessParameters(  
    this.OnGameSession,  
    this.OnProcessTerminate,  
    this.OnHealthCheck,  
    this.OnGameSessionUpdate,  
    port,  
    new LogParameters(logPaths));
```

3. Geben Sie das [ProcessParameters](#) Objekt an, wenn Sie [ProcessReady\(\)](#) aufrufen.

```
var processReadyOutcome =  
    GameLiftServerAPI.ProcessReady(processReadyParameter);
```

Ein vollständigeres Beispiel finden Sie unter [ProcessReady\(\)](#).

In Logs schreiben

Ihre Protokolldateien existieren, nachdem Ihr Serverprozess gestartet wurde. Sie können mit jeder Methode in die Protokolle schreiben, um in Dateien zu schreiben. Um die gesamte Standardausgabe und Fehlerausgabe Ihres Servers zu erfassen, ordnen Sie die Ausgabestreams den Protokolldateien neu zu, wie in den folgenden Beispielen:

C++

```
std::freopen("serverOut.log", "w+", stdout);
```

```
std::freopen("serverErr.log", "w+", stderr);
```

C#

```
Console.SetOut(new StreamWriter("serverOut.txt"));  
Console.SetError(new StreamWriter("serverErr.txt"));
```

Zugreifen auf Serverprotokolle

Der Protokollzugriff variiert je nach Flottenart:

Verwaltete EC2-Flotten

Wenn eine Spielsitzung endet, Amazon GameLift Servers werden die Logs automatisch in einem S3-Bucket gespeichert und 14 Tage lang aufbewahrt. Um den Speicherort der Protokolle für eine Spielsitzung zu ermitteln, können Sie den [GetGameSessionLogUrl](#) API-Vorgang verwenden. Verwenden Sie zum Herunterladen der Protokolle die URL, die der Vorgang zurückgibt.

Alternativ kannst du deine eigene Logging-Lösung einrichten, indem du deinen Spieleserver so konfigurierst, dass Logs direkt an deinen bevorzugten Logging-Service oder Speicherort gesendet werden. Weitere Informationen finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).

Containerflotten

Containerflotten erfassen Standardausgabe- und Fehlerströme aus allen Containern. Sie können die Protokollierung mit einer der folgenden Optionen konfigurieren:

- Speichern Sie die Containerausgabe als CloudWatch Protokollstreams in einer angegebenen Protokollgruppe
- Speichern Sie die Container-Ausgabe in einem S3-Speicher-Bucket
- Schalten Sie die Protokollierung aus (die Container-Ausgabe wird nicht gespeichert)

Ausführliche Informationen zur Konfiguration der Logging-Optionen bei der Erstellung von Containerflotten finden Sie [LogConfiguration](#) in der Amazon GameLift Servers API-Referenz.

Sicherheit in Amazon GameLift Servers

Wenn Sie Amazon EC2 Amazon GameLift Servers FleetIQ als eigenständige Funktion verwenden, finden Sie weitere Informationen unter [Sicherheit in Amazon EC2 im Amazon EC2](#) EC2-Benutzerhandbuch.

Cloud-Sicherheit hat höchste AWS Priorität. Als AWS -Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die eingerichtet wurden, um die Anforderungen der anspruchsvollsten Organisationen in puncto Sicherheit zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung zwischen 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 gelten Amazon GameLift Servers, finden Sie unter [AWS Leistungen im Umfang nach Compliance-Programmen AWS](#) .
- **Sicherheit in der Cloud** — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, darunter die Sensibilität Ihrer Daten, die Anforderungen Ihres Unternehmens AWS und die geltenden Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der geteilten Verantwortung bei der Verwendung von Amazon GameLift Servers einsetzen können. Die folgenden Themen veranschaulichen, wie Sie Amazon GameLift Servers zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren können. Sie lernen auch, wie Sie andere AWS Dienste nutzen können, die Ihnen bei der Überwachung und Sicherung Ihrer Amazon GameLift Servers Ressourcen helfen.

Topics

- [Datenschutz in Amazon GameLift Servers](#)
- [Identity and Access Management für Amazon GameLift Servers](#)
- [Protokollierung und Überwachung mit Amazon GameLift Servers](#)
- [Compliance-Validierung für Amazon GameLift Servers](#)
- [Resilienz in Amazon GameLift Servers](#)

- [Infrastruktursicherheit in Amazon GameLift Servers](#)
- [Konfigurations- und Schwachstellenanalyse in Amazon GameLift Servers](#)
- [Bewährte Methoden für die Sicherheit für Amazon GameLift Servers](#)

Datenschutz in Amazon GameLift Servers

Wenn Sie die Funktion Amazon GameLift Servers FleetIQ als eigenständige Funktion mit Amazon verwenden EC2, finden Sie weitere Informationen unter [Sicherheit bei Amazon EC2 im EC2 Amazon-Benutzerhandbuch](#).

Das AWS [Modell](#) der gilt für den Datenschutz in Amazon GameLift Servers. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. 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 der Konsole, der Amazon GameLift Servers API oder auf andere AWS-Services Weise arbeiten oder diese verwenden. AWS CLI AWS SDKs 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.

Amazon GameLift Servers-spezifische Daten werden wie folgt behandelt:

- Spielserver-Builds und Skripte, auf die Sie hochladen, Amazon GameLift Servers werden in Amazon S3 gespeichert. Nach dem Hochladen ist kein direkter Zugriff des Kunden auf diese Daten mehr möglich. Ein autorisierter Benutzer kann temporären Zugriff erhalten, um Dateien hochzuladen, kann die Dateien jedoch nicht direkt in Amazon S3 anzeigen oder aktualisieren. Um Skripte und Builds zu löschen, verwenden Sie die Amazon GameLift Servers-Konsole oder die Service-API.
- Die Protokolldaten der Spielsitzungen werden nach Abschluss der Spielsitzung für einen begrenzten Zeitraum in Amazon S3 gespeichert. Autorisierte Benutzer können auf die Protokolldaten zugreifen, indem sie sie über einen Link in der Amazon GameLift Servers-Konsole oder durch Aufrufe der Service-API herunterladen.
- Metrik- und Ereignisdaten werden in Amazon GameLift Servers gespeichert und sind über die Amazon GameLift Servers-Konsole oder durch Aufrufe der Service-API zugänglich. Es können Daten über Flotten, Instances, Platzierungen in Spielsitzungen, Matchmaking-Tickets, Spielsitzungen und Spielersitzungen abgerufen werden. Auf Daten kann auch über Amazon CloudWatch und CloudWatch Events zugegriffen werden.
- Vom Kunden bereitgestellte Daten werden in Amazon GameLift Servers gespeichert. Autorisierte Benutzer können über Aufrufe der Service-API darauf zugreifen. Potenziell sensible Daten sind etwa Spielerdaten, Spielersitzungs- und Spielsitzungsdaten (einschließlich Verbindungsinformationen), Matchmaker-Daten usw.

 Note

Wenn Sie IDs in Ihren Anfragen benutzerdefinierte Spieler angeben, wird erwartet, dass diese Werte anonymisiert sind UUIDs und keine identifizierenden Spielerinformationen enthalten.

Weitere Informationen zum Datenschutz enthält der Blog-Beitrag [AWS Shared Responsibility Model and GDPR](#) im AWS -Sicherheitsblog.

Verschlüsselung im Ruhezustand

Die Verschlüsselung im Ruhezustand von Amazon GameLift Servers-spezifischen Daten wird wie folgt behandelt:

- Spielserver-Builds und -Skripts werden in Amazon S3 S3-Buckets mit serverseitiger Verschlüsselung gespeichert.
- Vom Kunden bereitgestellte Daten werden in Amazon GameLift Servers in einem verschlüsselten Format gespeichert.

Verschlüsselung während der Übertragung

Verbindungen zu den Amazon GameLift Servers APIs werden über eine sichere (SSL) Verbindung hergestellt und mit [AWS Signature Version 4](#) authentifiziert (wenn eine Verbindung über die AWS CLI oder das AWS SDK hergestellt wird, erfolgt die Signierung automatisch). Die Authentifizierung wird mithilfe der IAM-definierten Zugriffsrichtlinien für die Sicherheitsanmeldeinformationen verwaltet, die zum Herstellen der Verbindung verwendet werden.

Die direkte Kommunikation zwischen Spiele-Clients und Spieleservern geschieht wie folgt:

- Für benutzerdefinierte Spieleserver, die auf Amazon GameLift Servers-Ressourcen gehostet werden, ist der Amazon GameLift Servers-Service nicht an der Kommunikation beteiligt. Die Verschlüsselung dieser Kommunikation liegt in der Verantwortung des Kunden. Sie können TLS-fähige Flotten verwenden, damit Ihre Spiele-Clients den Spieleserver bei der Verbindung authentifizieren und die gesamte Kommunikation zwischen Ihrem Spiele-Client und dem Spieleserver verschlüsseln.

- Zertifikate für TLS-fähige Flotten werden gleichzeitig mit der Flotte erstellt, und ihre Gültigkeitsdaten basieren auf dem Erstellungsdatum der Flotte.
- Denn wenn die Amazon GameLift Servers Realtime Generierung von TLS-Zertifikaten aktiviert ist, wird der Datenverkehr zwischen dem Spielclient und den Realtime Servern, für die das Client-SDK verwendet Realtime wird, während der Übertragung verschlüsselt. TCP-Datenverkehr wird mit TLS 1.2 verschlüsselt und UDP-Datenverkehr wird mit DTLS 1.2 verschlüsselt.

Note

Um die Anforderungen der Zertifizierungsstelle für TLS-Zertifikate zu erfüllen, Amazon GameLift Servers passt die maximale Gültigkeitsdauer von Zertifikaten für Flotten an, die für die Generierung eines Zertifikats konfiguriert sind. Die Gültigkeitsdauer von Zertifikaten beginnt mit der Erstellung der Flotte und wird nach dem folgenden Zeitplan geändert:

- Bis zum 11. März 2026 beträgt die maximale Gültigkeitsdauer für ein ausgestelltes TLS-Zertifikat 398 Tage.
- Ab dem 1. März 2026 beträgt die maximale Lebensdauer eines ausgestellten TLS-Zertifikats 200 Tage.
- Ab dem 1. März 2027 beträgt die maximale Lebensdauer eines ausgestellten TLS-Zertifikats 100 Tage.
- Ab dem 1. März 2029 beträgt die maximale Lebensdauer eines ausgestellten TLS-Zertifikats 47 Tage.

Um sicherzustellen, dass Ihre Zertifikate erneuert werden, und um die Laufzeitumgebungen der up-to-date Spieleserver aufrechtzuerhalten, Amazon GameLift Servers empfiehlt es sich, Ihre Spieleserverflotten regelmäßig auszutauschen.

Richtlinie für den Datenverkehr zwischen Netzwerken

Sie können aus der Ferne auf Ihre Amazon GameLift Servers-Instances zugreifen. Für Instances, die Linux verwenden, stellt SSH einen sicheren Kommunikationskanal für den Fernzugriff bereit. Verwenden Sie für Instances, auf denen Windows ausgeführt wird, einen RDP (Remote Desktop Protocol)-Client. Mit Amazon GameLift Servers FleetIQ wird der Fernzugriff auf Ihre Instances mithilfe von AWS Systems Manager Session Manager und Run Command mit TLS 1.2 verschlüsselt, und Anfragen zum Herstellen einer Verbindung werden mit Sigv4 signiert. Hilfe zum Herstellen einer

Verbindung mit einer verwalteten Amazon GameLift Servers-Instance finden Sie unter [Stellen Sie eine Remoteverbindung zu Amazon GameLift Servers Flotteninstanzen her](#).

Identity and Access Management für Amazon GameLift Servers

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 (Berechtigungen besitzt) ist, um Amazon GameLift Servers Ressourcen zu nutzen. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- [Zielgruppe](#)
- [Authentifizierung mit Identitäten](#)
- [Verwalten des Zugriffs mit Richtlinien](#)
- [Funktionsweise von Amazon GameLift Servers mit IAM](#)
- [Beispiele für identitätsbasierte Richtlinien für Amazon GameLift Servers](#)
- [Fehlerbehebung für Amazon GameLift Servers-Identität und -Zugriff](#)
- [AWS verwaltete Richtlinien für Amazon GameLift Servers](#)

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe [Fehlerbehebung für Amazon GameLift Servers-Identität und -Zugriff](#)).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe [Funktionsweise von Amazon GameLift Servers mit IAM](#)).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe [Beispiele für identitätsbasierte Richtlinien für Amazon GameLift Servers](#)).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter [So melden Sie sich bei Ihrem AWS-Konto an](#) im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter [AWS Signature Version 4 for API requests](#) im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter [Tasks that require root user credentials](#) im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter [Was ist IAM Identity Center?](#) im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein [IAM-Benutzer](#) ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle

von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter [Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu](#) können.

Eine [IAM-Gruppe](#) spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter [Anwendungsfälle für IAM-Benutzer](#) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine [IAM-Rolle](#) ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie [von einer Benutzer- zu einer IAM-Rolle \(Konsole\) wechseln](#) AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter [Methoden, um eine Rolle zu übernehmen](#) im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter [Übersicht über JSON-Richtlinien](#) im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcen und unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten 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.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter [Choose between managed policies and inline policies](#) im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen 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. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#).

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- **Berechtigungsgrenzen** – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter [Berechtigungsgrenzen für IAM-Entitäten](#) im -IAM-Benutzerhandbuch.
- **Richtlinien zur Dienstkontrolle (SCPs)** — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter [Service-Kontrollrichtlinien](#) im AWS Organizations -Benutzerhandbuch.
- **Richtlinien zur Ressourcenkontrolle (RCPs)** — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter [Richtlinien zur Ressourcenkontrolle \(RCPs\)](#).

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter [Sitzungsrichtlinien](#) im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die daraus resultierenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter [Bewertungslogik für Richtlinien](#) im IAM-Benutzerhandbuch.

Funktionsweise von Amazon GameLift Servers mit IAM

Bevor Sie IAM zum Verwalten des Zugriffs auf Amazon GameLift Servers verwenden, erfahren Sie, welche IAM-Funktionen Sie mit Amazon GameLift Servers verwenden können.

IAM-Funktionen, die Sie mit Amazon GameLift Servers verwenden können

IAM-Feature	Amazon GameLift Servers-Support
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (servicespezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja
Prinzipalberechtigungen	Ja
Servicerollen	Ja

IAM-Feature	Amazon GameLift Servers-Support
Service-verknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie Amazon GameLift Servers und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im [AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren](#).

Identitätsbasierte Richtlinien für Amazon GameLift Servers

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.

Beispiele für identitätsbasierte Richtlinien für Amazon GameLift Servers

Beispiele für identitätsbasierte Amazon GameLift Servers-Richtlinien finden Sie unter [Beispiele für identitätsbasierte Richtlinien für Amazon GameLift Servers](#).

Ressourcenbasierte Richtlinien in Amazon GameLift Servers

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 GameLift Servers

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 GameLift Servers Aktionen finden Sie unter [Aktionen definiert von Amazon GameLift Servers](#) in der Serviceautorisierungsreferenz.

Richtlinienaktionen in Amazon GameLift Servers verwenden das folgende Präfix vor der Aktion:

```
gamelift
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [  
  "gamelift:action1",  
  "gamelift:action2"  
]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "gamelift:Describe*"
```

Beispiele für identitätsbasierte Amazon GameLift Servers-Richtlinien finden Sie unter [Beispiele für identitätsbasierte Richtlinien für Amazon GameLift Servers](#).

Richtlinienressourcen für Amazon GameLift Servers

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": "*" 
```

Eine Liste der Amazon GameLift Servers Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter [Resources defined by Amazon GameLift Servers](#) in der Service Authorization Reference. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter [Von Amazon GameLift Servers definierte Aktionen](#).

Einige Amazon GameLift Servers Ressourcen haben ARN-Werte, sodass der Zugriff der Ressourcen mithilfe von IAM-Richtlinien verwaltet werden kann. Die Amazon GameLift Servers-Flottenressource hat einen ARN mit der folgenden Syntax:

```
arn:${Partition}:gamelift:${Region}:${Account}:fleet/${FleetId}
```

Weitere Informationen zum Format von ARNs finden Sie unter [Amazon Resource Names \(ARNs\)](#) in der Allgemeine AWS-Referenz.

Um beispielsweise die `fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa`-Flotte in Ihrer Anweisung anzugeben, verwenden Sie den folgenden ARN.

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

Um alle Flotten anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie einen Platzhalter (*):

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/*"
```

Beispiele für identitätsbasierte Amazon GameLift Servers-Richtlinien finden Sie unter [Beispiele für identitätsbasierte Richtlinien für Amazon GameLift Servers](#).

Richtlinien-Bedingungsschlüssel für Amazon GameLift Servers

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: 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 `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.

Eine Liste der Amazon GameLift Servers Bedingungsschlüssel finden Sie unter [Bedingungsschlüssel für Amazon GameLift Servers](#) in der Service Authorization Reference. Informationen dazu, mit welchen Aktionen und Ressourcen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter [Von Amazon GameLift Servers definierte Aktionen](#).

Beispiele für identitätsbasierte Amazon GameLift Servers-Richtlinien finden Sie unter [Beispiele für identitätsbasierte Richtlinien für Amazon GameLift Servers](#).

ACLs in Amazon GameLift Servers

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 GameLift Servers

Unterstützt ABAC (Tags in Richtlinien): Ja

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.

Ein Beispiel für eine identitätsbasierte Richtlinie, die den Zugriff auf eine Ressource anhand der Tags auf dieser Ressource beschränkt, finden Sie unter. [Amazon GameLift Servers-Flotten basierend auf Tags anzeigen](#)

Verwenden temporärer Anmeldeinformationen mit Amazon GameLift Servers

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 [Temporary security credentials in IAM](#) und [AWS-Services that work with IAM](#) im IAM-Benutzerhandbuch.

Serviceübergreifende Prinzipal-Berechtigungen für Amazon GameLift Servers

Unterstützt Forward Access Sessions (FAS): Ja

Forward Access Sessions (FAS) verwenden die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen AWS-Service an nachgeschaltete Dienste

zu stellen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter [Zugriffssitzungen weiterleiten](#).

Servicerollen für Amazon GameLift Servers

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 [Rollen zum Delegieren von Berechtigungen an einen AWS-Service erstellen](#) im IAM-Benutzerhandbuch.

Warning

Die Änderung der Berechtigungen für eine Servicerolle kann die Funktionalität von Amazon GameLift Servers beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Amazon GameLift Servers eine Anleitung dazu gibt.

Erlauben Sie Ihren Amazon GameLift Servers -gehosteten Spieleservern den Zugriff auf andere AWS Ressourcen, wie z. B. eine AWS Lambda Funktion oder eine Amazon DynamoDB Datenbank. Da Spieleserver auf Flotten gehostet werden, die Amazon GameLift Servers verwaltet werden, benötigen Sie eine Servicerolle, die Amazon GameLift Servers eingeschränkten Zugriff auf Ihre anderen Ressourcen gewährt. AWS Weitere Informationen finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).

Serviceverknüpfte Rollen für Amazon GameLift Servers

Unterstützt serviceverknüpfte Rollen: Ja

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft 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.

Einzelheiten zum Erstellen oder Verwalten von dienstbezogenen Rollen finden Sie im [AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren](#). Suchen Sie in der Tabelle nach einem Dienst, der in der Spalte mit dienstverknüpften **Yes** Rollen eine enthält. Wählen Sie Ja, um die Dokumentation der dienstbezogenen Rolle für diesen Dienst anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Amazon GameLift Servers

Benutzer und Rollen haben standardmäßig nicht die Berechtigung, Amazon GameLift Servers-Ressourcen zu erstellen oder zu ändern. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter [Erstellen von IAM-Richtlinien \(Konsole\)](#) im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von definiert wurden Amazon GameLift Servers, einschließlich des Formats von ARNs für jeden der Ressourcentypen, finden Sie unter [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon GameLift Servers](#) in der Referenz zur Serviceautorisierung.

Themen

- [Best Practices für Richtlinien](#)
- [Verwenden der Amazon GameLift Servers-Konsole](#)
- [Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer](#)
- [Erlauben des Spielerzugriffs für Spielsitzungen](#)
- [Erlaubt den Zugriff auf eine Amazon GameLift Servers Warteschlange](#)
- [Amazon GameLift Servers-Flotten basierend auf Tags anzeigen](#)
- [Greifen Sie in Amazon S3 auf eine Spiele-Build-Datei zu](#)

Best Practices für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Amazon GameLift Servers-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen

finden Sie unter [Von AWS verwaltete Richtlinien](#) oder [Von AWS verwaltete Richtlinien für Auftragsfunktionen](#) im IAM-Benutzerhandbuch.

- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter [Richtlinien und Berechtigungen in IAM](#) im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter [IAM-JSON-Richtlinienelemente: Bedingung](#) im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter [Richtlinienvvalidierung mit IAM Access Analyzer](#) im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter [Sicherer API-Zugriff mit MFA](#) im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter [Best Practices für die Sicherheit in IAM](#) im IAM-Benutzerhandbuch.

Verwenden der Amazon GameLift Servers-Konsole

Um auf die Amazon GameLift Servers-Konsole zuzugreifen, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen Ihnen das Auflisten und Anzeigen

von Details zu den Amazon GameLift Servers Ressourcen in Ihrem ermöglichen. AWS-Konto Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Um sicherzustellen, dass diese Entitäten die Amazon GameLift Servers Konsole weiterhin verwenden können, fügen Sie Benutzern und Gruppen Berechtigungen mit der Syntax in den folgenden Beispielen und in hinzu [Beispiele für Administratorberechtigungen](#). Weitere Informationen finden Sie unter [Legen Sie Benutzerberechtigungen fest für Amazon GameLift Servers](#).

Benutzer, die mit Amazon GameLift Servers Über AWS CLI - oder AWS API-Operationen arbeiten, benötigen keine Mindestberechtigungen für die Konsole. Stattdessen können Sie den Zugriff auf die Operationen beschränken, die der Benutzer ausführen muss. Beispielsweise benötigt ein Spielerbenutzer, der im Namen von Spieleclients handelt, Zugriff, um Spielsitzungen anzufordern, Spieler in Spiele einzuordnen und andere Aufgaben zu erledigen.

Informationen zu den Berechtigungen, die für die Nutzung aller Amazon GameLift Servers Konsolenfunktionen erforderlich sind, finden Sie unter Berechtigungssyntax für Administratoren in [Beispiele für Administratorberechtigungen](#).

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der AWS CLI AWS OR-API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    }
  ]
}
```



```

    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}

```

Erlauben des Spielerzugriffs für Spielsitzungen

Um Spieler an Spielsitzungen teilnehmen zu lassen, benötigen Spielclients und Backend-Dienste Berechtigungen. Richtlinienbeispiele für diese Szenarien finden Sie unter [Beispiele für Benutzerberechtigungen von Spielern](#).

Erlaubt den Zugriff auf eine Amazon GameLift Servers Warteschlange

Das folgende Beispiel bietet einem Benutzer Zugriff auf eine bestimmte Amazon GameLift Servers Warteschlange.

Diese Richtlinie gewährt dem Benutzer Berechtigungen zum Hinzufügen, Aktualisieren und Löschen von Warteschlangenzielen mit den folgenden Aktionen: `gamelift:UpdateGameSessionQueue`, `gamelift>DeleteGameSessionQueue`, und `gamelift:DescribeGameSessionQueues`. Wie gezeigt, verwendet diese Richtlinie das Resource Element, um den Zugriff auf eine einzelne Warteschlange zu beschränken: `gamesessionqueue/examplequeue123`.

JSON

```

{
  "Version": "2012-10-17",

```

```

"Statement": [
  {
    "Sid": "ViewSpecificQueueInfo",
    "Effect": "Allow",
    "Action": [
      "gamelift:DescribeGameSessionQueues"
    ],
    "Resource": "arn:aws:gamelift:us-east-1:555555555555:gamesessionqueue/
examplequeue123"
  },
  {
    "Sid": "ManageSpecificQueue",
    "Effect": "Allow",
    "Action": [
      "gamelift:UpdateGameSessionQueue",
      "gamelift>DeleteGameSessionQueue"
    ],
    "Resource": "arn:aws:gamelift:us-east-1:111122223333:gamesessionqueue/
examplequeue123"
  }
]
}

```

Amazon GameLift Servers-Flotten basierend auf Tags anzeigen

Sie können in Ihrer identitätsbasierten Richtlinie Bedingungen für die Steuerung des Zugriffs auf Amazon GameLift Servers-Ressourcen auf der Basis von Tags verwenden. Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen können, die das Anzeigen einer Flotte ermöglicht, wenn das Owner Tag mit dem Benutzernamen des Benutzers übereinstimmt. Diese Richtlinie gewährt auch die erforderlichen Berechtigungen, um diesen Vorgang in der Konsole abzuschließen.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListFleetsInConsole",
      "Effect": "Allow",
      "Action": "gamelift:ListFleets",
      "Resource": "*"
    }
  ]
}

```

```

    },
    {
      "Sid": "ViewFleetIfOwner",
      "Effect": "Allow",
      "Action": "gamelift:DescribeFleetAttributes",
      "Resource": "arn:aws:gamelift:*:*:fleet/*",
      "Condition": {
        "StringEquals": {"aws:ResourceAccount": "${aws:username}"}
      }
    }
  ]
}

```

Greifen Sie in Amazon S3 auf eine Spiele-Build-Datei zu

Nachdem Sie Ihren Spielserver mit integriert haben Amazon GameLift Servers, laden Sie die Build-Dateien auf Amazon S3 hoch. Verwenden Amazon GameLift Servers Sie die folgende Richtlinie, um auf die Build-Dateien zuzugreifen.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::bucket-name/object-name"
    }
  ]
}

```

Weitere Informationen zum Hochladen von Amazon GameLift Servers Spieldateien finden Sie unter [Erstelle einen Gameserver-Build für Amazon GameLift Servers](#).

Fehlerbehebung für Amazon GameLift Servers-Identität und -Zugriff

Mithilfe der folgenden Informationen können Sie häufig auftretende Probleme diagnostizieren und beheben, die bei der Arbeit mit Amazon GameLift Servers und AWS Identity and Access Management (IAM) auftreten können.

Themen

- [Ich bin nicht autorisiert, eine Aktion in Amazon GameLift Servers auszuführen.](#)
- [Ich bin nicht berechtigt, iam auszuführen: PassRole](#)
- [Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Amazon GameLift Servers Ressourcen ermöglichen](#)

Ich bin nicht autorisiert, eine Aktion in Amazon GameLift Servers auszuführen.

Wenn Ihnen AWS-Managementkonsole mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion durchzuführen, wenden Sie sich an Ihren AWS Kontoadministrator, um Unterstützung zu erhalten. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Der folgende Beispielfehler tritt auf, wenn der mateojackson IAM-Benutzer versucht, die Konsole zu verwenden, um Details zu einer Warteschlange anzuzeigen, aber nicht über die `gamelift:DescribeGameSessionQueues` entsprechenden Berechtigungen verfügt:

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
gamelift:DescribeGameSessionQueues on resource: examplequeue123
```

In diesem Fall bittet Mateo seinen Administrator, seine Richtlinien zu aktualisieren, um ihm Lesezugriff auf die `examplequeue123` Ressource zu gewähren, die die Aktion verwendet. `gamelift:DescribeGameSessionQueues`

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Amazon GameLift Servers übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon GameLift Servers auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Amazon GameLift Servers Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob Amazon GameLift Servers diese Features unterstützt, finden Sie unter [Funktionsweise von Amazon GameLift Servers mit IAM](#).
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen in AWS-Konten Ihrem Besitz gewähren können, finden Sie im IAM-Benutzerhandbuch unter [Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto, dem Sie](#) gehören.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie [AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte](#).
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter [Gewähren von Zugriff für extern authentifizierte Benutzer \(Identitätsverbund\)](#) im IAM-Benutzerhandbuch.

- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinien für Amazon GameLift Servers

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. 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 für alle AWS Kunden verfügbar sind. 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 Prinzipalidentitäten (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.

AWS verwaltete Richtlinie: GameLiftContainerFleetPolicy

Sie können Rollen `GameLiftContainerFleetPolicy` an Ihre IAM-Rollen anhängen.

Die Richtlinie gewährt Berechtigungen für Rechenaktionen in einer Amazon GameLift Servers Containerflotte. Eine Containerflotte besteht aus einer Reihe von Hosting-Ressourcen, die für Sie Amazon GameLift Servers verwaltet werden. Amazon GameLift Servers benötigt Berechtigungen, um in Ihrem Namen eine Verbindung mit dem Amazon GameLift Servers Dienst und anderen AWS Diensten herzustellen.

Wenn Sie eine Containerflotte mit erstellen Amazon GameLift Servers, geben Sie eine IAM-Servicerolle mit der angehängten `GameLiftContainerFleetPolicy` verwalteten Richtlinie an.

Anweisungen zum Erstellen der Servicerolle finden Sie unter [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#).

Weitere Informationen finden Sie unter [GameLiftContainerFleetPolicy](#).

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `cloudwatch`— Ermöglicht Amazon GameLift Servers das Schreiben von Spielsitzungsprotokollen in einen Amazon CloudWatch Events-Protokollstream in Ihrem AWS Konto.
- `cloudwatch` — Ermöglicht Amazon GameLift Servers die Erstellung einer CloudWatch Protokollgruppe, um Spielsitzungsdaten im Protokollstream zu organisieren.
- `s3`— Ermöglicht Amazon GameLift Servers das Schreiben von Spielsitzungsprotokollen in einen Amazon Simple Storage Service-Bucket in Ihrem AWS Konto.
- `s3`— Ermöglicht Amazon GameLift Servers das Abrufen des AWS-Region Standorts eines bestimmten Amazon S3 S3-Buckets mithilfe der API-Aktions`s3:GetBucketLocation`.
- `gamelift` — Ermöglicht Amazon GameLift Servers das Abrufen eines Authentifizierungstokens, das es einem gehosteten Spieleserver ermöglicht, über Ihr AWS Konto mit dem Amazon GameLift Servers Dienst zu kommunizieren.

Amazon GameLift ServersAktualisierungen der AWS verwalteten Richtlinien

Hier finden Sie Informationen zu Aktualisierungen AWS verwalteter Richtlinien, die Amazon GameLift Servers seit Beginn der Nachverfolgung dieser Änderungen durch diesen Dienst vorgenommen wurden. Abonnieren Sie den RSS-Feed auf der Seite mit den [Amazon GameLift ServersVersionshinweisen, um automatische Benachrichtigungen über Änderungen an dieser Seite](#) zu erhalten.

Änderungen	Beschreibung	Date
GameLiftContainerFleetPolicy — Änderung	Amazon GameLift Serversneue Berechtigungen zum Abrufen AWS-Region eines Amazon S3 S3-Buckets hinzugefügt.	5. Februar 2024
GameLiftContainerFleetPolicy – Neue Richtlinie	Amazon GameLift Serversneue Berechtigungen hinzugefügt, damit Gameserver-Contain	12. November 2024

Änderungen	Beschreibung	Date
	er auf Amazon GameLift Servers verwalteten Flotten ausgeführt werden können.	
Amazon GameLift Servers hat die Änderungsverfolgung gestartet	Amazon GameLift Servers hat begonnen, Änderungen an den AWS verwalteten Richtlinien zu verfolgen.	12. November 2024

Protokollierung und Überwachung mit Amazon GameLift Servers

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon GameLift Servers und Ihre AWS Lösungen. Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen etwaigen Ausfall an mehreren Stellen leichter debuggen können.

AWS und Amazon GameLift Servers stellen mehrere Tools zur Verfügung, mit denen Sie Ihre Spielhosting-Ressourcen überwachen und auf mögliche Vorfälle reagieren können.

CloudWatch Amazon-Alarme

Mithilfe von CloudWatch Amazon-Alarmen beobachten Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum. Wenn die Metrik einen bestimmten Schwellenwert überschreitet, wird eine Benachrichtigung an ein Amazon SNS Thema oder eine AWS Auto Scaling Scaling-Richtlinie gesendet. CloudWatch Alarme werden ausgelöst, wenn sich ihr Status ändert, und sie werden für eine bestimmte Anzahl von Zeiträumen aufrechterhalten, nicht wenn sich ein bestimmter Zustand befindet. Weitere Informationen finden Sie unter [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#).

AWS CloudTrail Logs

CloudTrail bietet eine Aufzeichnung der Aktionen, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt wurden Amazon GameLift Servers. Anhand der von CloudTrail gesammelten Informationen können Sie feststellen, welche Anfrage gestellt wurde Amazon GameLift Servers, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details. Weitere Informationen finden Sie unter [Amazon GameLift ServersAPI-Aufrufe protokollieren mit AWS CloudTrail](#).

Compliance-Validierung für Amazon GameLift Servers

Amazon GameLift Servers fällt nicht in den Geltungsbereich AWS irgendwelcher Compliance-Programme.

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](#).

Resilienz in Amazon GameLift Servers

Wenn du verwendest Amazon GameLift Servers FleetIQ als eigenständige Funktion bei Amazon EC2, siehe [Sicherheit bei Amazon EC2](#) im EC2 Amazon-Benutzerhandbuch.

Die AWS globale Infrastruktur basiert auf AWS Regionen und 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 AWS Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Zusätzlich zur AWS globalen Infrastruktur Amazon GameLift Servers bietet die folgenden Funktionen zur Unterstützung Ihrer Anforderungen an die Datenstabilität:

- Warteschlangen mit mehreren Regionen — Amazon GameLift Servers Warteschlangen für Spielsitzungen werden verwendet, um neue Spielsitzungen mit verfügbaren Hosting-Ressourcen

zu platzieren. Warteschlangen, die sich über mehrere Regionen erstrecken, können Spielsitzungen im Falle eines regionalen Ausfalls umleiten. Weitere Informationen und bewährte Methoden zum Erstellen von Spielsitzungswarteschlangen finden Sie unter [Passen Sie eine Warteschlange für Spielsitzungen an](#).

- Automatische Kapazitätsskalierung — Sorgen Sie für den Zustand und die Verfügbarkeit Ihrer Hosting-Ressourcen, indem Sie Amazon GameLift Servers Tools zur Skalierung. Diese Tools bieten eine Reihe von Optionen, mit denen Sie die Flottenkapazität an die Bedürfnisse Ihres Spiels und Ihrer Spieler anpassen können. Weitere Informationen zur Skalierung finden Sie unter [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#).
- Verteilung auf mehrere Instanzen — Amazon GameLift Servers verteilt den eingehenden Traffic je nach Flottengröße auf mehrere Instanzen. Als bewährte Methode sollten Spiele in der Produktion über mehrere Instances verfügen, um die Verfügbarkeit für den Fall zu wahren, dass eine Instance fehlerhaft wird oder nicht mehr reagiert.
- Amazon S3 S3-Speicher — Spieleserver-Builds und -Skripte, die hochgeladen werden Amazon GameLift Servers werden in Amazon S3 mithilfe der Standard-Speicherklasse gespeichert, die mehrere Rechenzentrumsreplikationen verwendet, um die Ausfallsicherheit zu erhöhen. Spielsitzungsprotokolle werden auch in Amazon S3 unter Verwendung der Standard-Speicherklasse gespeichert.

Infrastruktursicherheit in Amazon GameLift Servers

Wenn du verwendest Amazon GameLift Servers FleetIQ als eigenständige Funktion bei Amazon EC2, siehe [Sicherheit bei Amazon EC2](#) im EC2 Amazon-Benutzerhandbuch.

Als verwalteter Service Amazon GameLift Servers ist durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper [Amazon Web Services: Sicherheitsprozesse im Überblick](#) beschrieben sind.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff Amazon GameLift Servers über das Netzwerk. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Wir empfehlen TLS 1.3 oder höher. Clients müssen außerdem Verschlüsselungssammlungen mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen. Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit [AWS](#)

[-Security-Token-Service](#) (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Das Tool Amazon GameLift Servers Der Service platziert alle Flotten in Amazon Virtual Private Clouds (VPCs), sodass sich jede Flotte in einem logisch isolierten Bereich in der AWS Cloud befindet. Sie können Folgendes verwenden ... Amazon GameLift Servers Richtlinien zur Steuerung des Zugriffs von bestimmten VPC-Endpunkten oder bestimmten VPCs. Dadurch wird der Netzwerkzugriff auf ein bestimmtes Objekt effektiv isoliert. Amazon GameLift Servers Ressource nur von der spezifischen VPC innerhalb des AWS Netzwerks. Wenn Sie eine Flotte erstellen, geben Sie einen Bereich von Portnummern und IP-Adressen an. Diese Bereiche begrenzen, wie eingehender Datenverkehr auf gehostete Spieleserver in einer Flotten-VPC zugreifen kann. Verwenden Sie bewährte Standardmethoden für die Sicherheit bei der Auswahl von Flottenzugriffseinstellungen.

Konfigurations- und Schwachstellenanalyse in Amazon GameLift Servers

Wenn Sie Amazon EC2 Amazon GameLift Servers FleetIQ als eigenständige Funktion verwenden, finden Sie weitere Informationen unter [Sicherheit in Amazon EC2 im Amazon EC2](#) EC2-Benutzerhandbuch.

Konfiguration und IT-Steuerelemente unterliegen der übergreifenden Verantwortlichkeit von AWS und Ihnen, unserem Kunden. Weitere Informationen finden Sie im Modell der AWS [gemeinsamen](#) Verantwortung. AWS kümmert sich um grundlegende Sicherheitsaufgaben wie das Patchen von Gastbetriebssystemen (OS) und Datenbanken, die Firewallkonfiguration und die Notfallwiederherstellung. Diese Verfahren wurden von qualifizierten Dritten überprüft und zertifiziert. Weitere Informationen finden Sie in der folgenden Ressource: [Amazon Web Services: Überblick über Sicherheitsprozesse](#) (Whitepaper).

Die folgenden bewährten Sicherheitsmethoden beinhalten auch die Adresskonfiguration und Schwachstellenanalyse in Amazon GameLift Servers:

- Kunden sind für die Verwaltung der Software verantwortlich, die für Amazon GameLift Servers-Instances für das Spiele-Hosting bereitgestellt wird. Das heißt:
 - Von Kunden bereitgestellte Spieleserver-Anwendungssoftware muss regelmäßig gewartet werden, einschließlich Updates und Sicherheitspatches. Um die Spielserver-Software zu aktualisieren, laden Sie einen neuen Build in ein neues Container-Image hoch Amazon GameLift Servers oder stellen Sie ihn in einem neuen Container-Image bereit, erstellen Sie eine neue Flotte dafür und leiten Sie den Traffic auf die neue Flotte um.

- Das Basis-AMI (Amazon Machine Image), das das Betriebssystem enthält, wird nur aktualisiert, wenn eine neue Flotte erstellt wird. Tauschen Sie regelmäßig Flotten aus, um das Betriebssystem und andere Anwendungen, die Teil des AMI sind, zu patchen, zu aktualisieren und zu sichern, unabhängig von Spielservers-Updates. Produktionsflotten sollten mindestens einmal pro Jahr ausgetauscht werden. Weitere Details finden Sie unter [Halten Sie die Runtime-Umgebungen Ihrer Flotte auf dem neuesten Stand](#).
- Kunden sollten erwägen, ihre Spiele regelmäßig mit den neuesten SDK-Versionen zu aktualisieren, einschließlich des AWS SDK, des Amazon GameLift Servers Server-SDK und des Amazon GameLift Servers Client-SDK für Echtzeitserver.

Bewährte Methoden für die Sicherheit für Amazon GameLift Servers

Wenn Sie Amazon EC2 Amazon GameLift Servers FleetIQ als eigenständige Funktion verwenden, finden Sie weitere Informationen unter [Sicherheit in Amazon EC2 im Amazon EC2](#) EC2-Benutzerhandbuch.

Amazon GameLift Servers enthält eine Reihe von Sicherheitsfeatures, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden stellen allgemeine Richtlinien und keine vollständige Sicherheitslösung dar. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Halten Sie die Runtime-Umgebungen Ihrer Flotte auf dem neuesten Stand

Amazon GameLift Server empfiehlt dringend, dass Sie verwaltete Flotten (einschließlich verwalteter EC2- und verwalteter Container-Flotten) regelmäßig austauschen, um sichere Laufzeitumgebungen für Ihre Spieleserver aufrechtzuerhalten. Flotten, die über einen längeren Zeitraum ohne Runtime-Updates laufen, können veraltete Abhängigkeiten und Sicherheitslücken enthalten, die deine Spieleserver gefährden könnten. Einzelheiten zur Aufteilung der Verantwortung für Software, die in Amazon GameLift Servers Flotten eingesetzt wird, finden Sie unter [Konfigurations- und Schwachstellenanalyse in Amazon GameLift Servers](#)

Die Laufzeitumgebung einer verwalteten Flotte wird durch ihre Amazon Machine Image (AMI) - Version bestimmt. Wenn eine neue Flotte erstellt wird, Amazon GameLift Servers weist sie der Flotte die neueste verfügbare AMI-Version zu, und alle Compute-Instances in dieser Flotte werden mit dieser Version bereitgestellt. Um die AMI-Version zu aktualisieren, müssen Sie eine neue Flotte

erstellen. Einzelheiten zu aktuellen AMI-Versionen finden Sie unter [Amazon GameLift Servers AMI-Versionen](#).

Empfohlene Vorgehensweisen:

- Überwachen Sie das Flottenalter und ersetzen Sie Flotten, die älter als 30 Tage sind — Sie können das Erstellungsdatum einer Flotte in der Amazon GameLift Servers Konsole verfolgen oder die CLI verwenden, um Flottenattribute abzurufen. Amazon GameLift Servers zeigt in der Konsole Warnungen für Flotten an, die älter als 90 Tage sind, und benachrichtigt Kontoinhaber per E-Mail für Flotten, die älter als ein Jahr sind.

 Note

Durch das Aktualisieren einer Flotte (z. B. durch die Verwendung von [UpdateFleetAttributes](#) oder [UpdateContainerFleet](#)) wird die AMI-Version nicht geändert. Sie müssen eine neue Flotte erstellen.

- Tauschen Sie Flotten regelmäßig je nach Sicherheitsstatus aus — Richten Sie einen regelmäßigen Zeitplan ein, um neue Flotten zu erstellen und alte Flotten außer Dienst zu stellen. Erwägen Sie, einen Service wie [Amazon Q](#) zu nutzen, um Ihren Spielcode mit der aktuellen AMI-Version zu überprüfen, Sicherheitsprobleme zu erkennen und Maßnahmen zur Behebung vorzuschlagen.
- Testen Sie Server-Builds vor der Bereitstellung mit den neuesten AMI-Versionen — Möglicherweise müssen Sie Ihren Server-Build ändern und auf ihn hochladen, Amazon GameLift Servers bevor Sie eine neue Flotte erstellen können.
- Verwalten Sie Flottenkontingente für Ihr AWS Konto — Sie können bei Bedarf Limiterhöhungen beantragen, um Ersatzflotten zu erstellen. Weitere Informationen finden Sie unter [Endpunkte und Kontingente von Amazon GameLift Servers](#).
- Erwägen Sie die Automatisierung des Flottenaustauschs — Sie können Prozesse automatisieren, um neue Flotten zu erstellen und Spieler-Traffic aus älteren Flotten zu migrieren. Beispiel:
 - Wird verwendet AWS CloudFormation , um die Erstellung und Verwaltung von Flotten zu automatisieren. Pflegen Sie Ihre Flottenkonfigurationen als CloudFormation Vorlagen und verwenden Sie sie, um Ressourcenstapel zu starten.
 - Nutzen Sie die Amazon GameLift Servers Alias-Funktion, um eine bestimmte Flotte IDs zu abstrahieren. Mithilfe von Flottenaliasen ist es einfach, den Spielerverkehr von einer bestehenden Flotte auf eine neue umzustellen, ohne dass die laufenden Spielsitzungen unterbrochen werden. Details hierzu finden Sie unter [Zusammenfassung einer Amazon GameLift Servers Flottenbezeichnung mit einem Alias](#).

- Verwenden Sie blue/green Implementierungsstrategien, um das Migrationsrisiko zu reduzieren und Ausfallzeiten zu vermeiden. Mit zwei identischen Produktionsumgebungen können Sie von einer vollständigen, produktionsähnlichen Testumgebung profitieren, mehr Kontrolle über den Migrationsprozess ausüben und sofortige Rollbacks sicherstellen.

Schützen Sie Ihre Portkonfigurationen

Wir empfehlen dringend, keine Ports zum Internet zu öffnen, da dies ein Sicherheitsrisiko darstellt. Die folgende Konfiguration öffnet beispielsweise einen Remote-Desktop-Port, über den jeder Benutzer im Internet auf die Instanz zugreifen kann:

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "0.0.0.0/0",
      "Protocol": "RDP",
      "ToPort": 3389
    }
  ]
}
```

Verwenden Sie stattdessen, [UpdateFleetPortSettings](#) um einen Port mit einer bestimmten IP-Adresse oder einem bestimmten Adressbereich zu öffnen, wie in diesem Beispiel gezeigt:

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "54.186.139.221/32",
      "Protocol": "TCP",
      "ToPort": 3389
    }
  ]
}
```

Zusätzliche Sicherheitsressourcen

Weitere Informationen darüber, wie Sie Ihre Nutzung Amazon GameLift Servers sicherer gestalten können, finden Sie im [AWS Well-Architected Tool Bereich Sicherheit](#). .

Amazon GameLift ServersPreisgestaltung und Kostenplanung

Das Verständnis der Amazon GameLift Servers Preisstruktur und die Implementierung kostengünstiger Strategien in Ihrer Game-Hosting-Lösung sind für die Optimierung Ihres Budgets für die Spieleinfrastruktur unerlässlich. Detaillierte Serviceinformationen, einschließlich beispielhafter Preisszenarien und Tools zur Preisgestaltung, finden Sie unter [Amazon GameLift ServersPreisgestaltung](#).

AWS Neukunden können Amazon GameLift Servers das kostenlose Kontingent bis zu 12 Monate lang kostenlos nutzen. Halten Sie sich an die Nutzungsgrenzen des kostenlosen Kontingents, um Gebühren zu vermeiden.

Preismodelle

Amazon GameLift Serversbietet verschiedene Preismodelle, die auf unterschiedliche Nutzungsszenarien und Geschäftsanforderungen zugeschnitten sind. Jedes Modell hat eine andere Kostenbasis. Sie können die Kosten kontrollieren, indem Sie die Kostenfaktoren in jedem Modell anpassen.

Verwaltetes Hosting

Verwaltete EC2 und verwaltete Containerflotten verwenden EC2 Amazon-Instances Amazon GameLift Servers zur Verwaltung. Wenn Sie das Spiele-Hosting einrichten, wählen Sie den Typ der zu verwendenden EC2 Amazon-Instance aus.

- Rechenkosten — Zahlen Sie für die stündliche Nutzung der EC2 Instance für das Hosten von Spielesitzungen. Zu den Kostenfaktoren für Instanzen gehören:
 - AWS Region
 - Instanzfamilie, Typ und Größe
 - Verwendung von Spot-Instances gegenüber On-Demand-Instances
 - Betriebssystem (verwaltete Containerflotten laufen nur unter Linux)
- Datenübertragung — Zahlen Sie für den Datenverkehr zwischen Spieleclients und gehosteten Spieleservern

Hosten Sie überall

Anywhere-Hosting läuft auf Game-Hosting-Ressourcen, die Sie bereitstellen und verwalten, sodass keine Rechenkosten anfallen. Stattdessen basieren die Kosten auf:

- Spielsitzungen — Die Bezahlung basiert auf der Anzahl der Spielsitzungen, die auf Anywhere-Computern durchgeführt wurden
- Verbindungsminuten — Zahlen Sie für die Verbindungsminuten des Serverprozesses

Verwenden Sie den Preisrechner für Amazon GameLift Servers, um die Kosten für verschiedene Amazon GameLift Servers Konfigurationen zu schätzen. Verwenden Sie den Kalkulator bei der Entwicklung Ihrer Game-Hosting-Lösung oder zur Prognose der Kosten für Konfigurationsänderungen. Weitere Informationen finden Sie unter [Amazon GameLift ServersPreisvoranschläge erstellen](#).

Generieren Sie Amazon GameLift Servers Preisschätzungen

Mit AWS Pricing Calculator können Sie [einen Kostenvoranschlag für erstellen Amazon GameLift Servers](#). Sie benötigen keine gründlichen AWS-Konto oder fundierten Kenntnisse AWS , um den Taschenrechner zu verwenden.

AWS Pricing Calculator Der Taschenrechner führt Sie durch die Entscheidungen, die sich auf die Servicekosten auswirken, um Ihnen eine Vorstellung davon zu geben, wie viel Ihr Spieleprojekt kosten Amazon GameLift Servers könnte. Wenn du dir noch nicht sicher bist, wie du es verwenden möchtest Amazon GameLift Servers, verwende die Standardwerte, um eine Schätzung zu erstellen. Bei der Planung der Produktionsnutzung kann Ihnen der Rechner dabei helfen, mögliche Szenarien zu testen und genauere Schätzungen zu erstellen.

Schätzen Sie Amazon GameLift Servers Managed Hosting ein

Diese Option bietet einen Kostenvoranschlag für das Hosten Ihrer Spiele auf Amazon GameLift Servers verwalteten Servern, einschließlich der Kosten für die Nutzung der Serverinstanzen und die Datenübertragung. Bei Amazon GameLift Servers verwaltetem Hosting fallen keine zusätzlichen Kosten für das FlexMatch Matchmaking an.

Wenn du Spieleserver in mehr als einer AWS Region oder auf mehr als einem Instance-Typ hostest oder dies planen möchtest, erstelle eine Schätzung für jede Region und jeden Instance-Typ.

Amazon GameLift Servers-Instances

In diesem Abschnitt können Sie die Art und Anzahl der Rechenressourcen abschätzen, die Sie für die Ausrichtung von Spielsitzungen für Ihre Spieler benötigen. Amazon GameLift Servers verwendet [Amazon Elastic Compute Cloud \(Amazon EC2\) -Instances](#) zur Verwaltung von Spieleservern. Amazon GameLift Servers stellen Sie eine Flotte von Instances mit einem bestimmten Instance-Typ und Betriebssystem bereit. Wenn Sie mehrere Flotten haben oder planen, diese zu haben, erstellen Sie für jede Flotte einen Kostenvoranschlag.

Öffnen Sie zunächst die [Amazon GameLift Servers Konfigurationsseite](#) von AWS Pricing Calculator. Fügen Sie eine Beschreibung hinzu, wählen Sie eine Region aus und wählen Sie dann Estimate Amazon GameLift Servers hosting (Instance + Data Transfer Out). Füllen Sie unter Amazon GameLift Servers Instanzen die folgenden Felder aus:

- Maximale Anzahl gleichzeitiger Spieler (Spitzen-CCU)

Dies ist die maximale Anzahl von Spielern, die sich gleichzeitig mit Ihren Spieleservern verbinden können. Dieses Feld gibt an, wie viel Hosting-Kapazität Amazon GameLift Servers benötigt wird, um die Spitzennachfrage der Spieler zu decken. Geben Sie die tägliche Spitzenzahl an Spielern ein, die Sie voraussichtlich mithilfe von Instances in der von Ihnen ausgewählten AWS Region hosten werden.

Wenn du beispielsweise 1.000 Spielern gleichzeitig ermöglichen möchtest, eine Verbindung zu deinem Spiel herzustellen, behalte den Standardwert von bei **1000**.

- Durchschnittliche CCU pro Stunde als Prozentsatz der täglichen CCU zu Spitzenzeiten

Dies ist die durchschnittliche Anzahl gleichzeitiger Spieler pro Stunde über einen Zeitraum von 24 Stunden. Wir verwenden diesen Wert, um abzuschätzen, wie viele Hosting-Kapazitäten für Ihre Spieler dauerhaft aufrechterhalten Amazon GameLift Servers werden müssen. Wenn Sie sich nicht sicher sind, welchen Prozentwert Sie verwenden sollen, behalten Sie den Standardwert **50** Prozent bei. Für Spiele mit stabiler Spielernachfrage empfehlen wir, einen Wert von **70** Prozent einzugeben.

Wenn Ihr Spiel beispielsweise eine durchschnittliche stündliche CCU von 6.000 und eine maximale CCU von 10.000 hat, geben Sie den Wert in Prozent ein. **60**

- Spielsitzungen pro Instanz

Dies ist die Anzahl der Spielsitzungen, die jede deiner Spielserver-Instanzen gleichzeitig hosten kann. Zu den Faktoren, die sich auf diese Anzahl auswirken können, gehören die

Ressourcenanforderungen Ihres Spieleservers, die Anzahl der Spieler, die in jeder Spielsitzung gehostet werden sollen, und die Leistungserwartungen der Spieler. Wenn Sie die Anzahl der gleichzeitigen Spielsitzungen für Ihr Spiel kennen, geben Sie diesen Wert ein. Sie können auch den Standardwert von **20** beibehalten.

- Spieler pro Spielsitzung

Dies ist die durchschnittliche Anzahl von Spielern, die eine Verbindung zu einer Spielsitzung herstellen, wie in Ihrem Spieldesign definiert. Wenn du Spielmodi mit unterschiedlicher Anzahl von Spielern hast, schätze die durchschnittliche Anzahl von Spielern pro Spielsitzung während deines gesamten Spiels. Der Standardwert ist **8**.

- Puffer im Leerlauf der Instanz%

Dies ist der Prozentsatz der ungenutzten Hosting-Kapazität, der für plötzliche Spitzenzeiten bei der Spielernachfrage reserviert werden muss. Die Puffergröße ist ein Prozentsatz der Gesamtzahl der Instances in einer Flotte. Der Standardwert ist **10** Prozent.

Beispiel: Bei einem Leerlaufpuffer von 20 Prozent unterhält eine Flotte, die Spieler mit 100 aktiven Instances unterstützt, 20 inaktive Instances.

- Spot-Instance%

Amazon GameLift ServersFlotten können eine Kombination aus On-Demand-Instances und Spot-Instances verwenden. Während On-Demand-Instances eine zuverlässigere Verfügbarkeit bieten, bieten Spot-Instances eine äußerst kosteneffiziente Alternative. Wir empfehlen die Verwendung einer Kombination, um sowohl die Kosteneinsparungen als auch die Verfügbarkeit zu optimieren. Informationen zur Amazon GameLift Servers Verwendung von Spot-Instances finden Sie unter [On-Demand-Instances im Vergleich zu Spot Instances](#).

Geben Sie in dieses Feld den Prozentsatz der Spot-Instances ein, die in einer Flotte verwaltet werden sollen. Wir empfehlen einen Spot-Instance-Prozentsatz zwischen 50 und 85 Prozent. Der Standardwert ist **50** Prozent.

Wenn Sie beispielsweise eine Flotte mit 100 Instances bereitstellen und einen **40** Prozentwert angeben, Amazon GameLift Servers werden 60 On-Demand-Instances und 40 Spot-Instances verwaltet.

- Instance-Typ

Amazon GameLift ServersFlotten können eine Reihe von EC2 Amazon-Instance-Typen verwenden, die sich in Bezug auf Rechenleistung, Arbeitsspeicher, Speicher und Netzwerkfähigkeit

unterscheiden. Wenn Sie eine Amazon GameLift Servers Flotte konfigurieren, wählen Sie einen Instance-Typ, der den Anforderungen Ihres Spiels am besten entspricht. Informationen zur Auswahl eines Instance-Typs mit Amazon GameLift Servers finden Sie unter [Wählen Sie Rechenressourcen für eine verwaltete Flotte](#).

Wenn Sie den Instance-Typ kennen, den Sie in Ihrer Amazon GameLift Servers Flotte verwenden oder verwenden möchten, wählen Sie diesen Typ aus. Wenn Sie sich nicht sicher sind, welchen Typ Sie wählen sollen, sollten Sie sich für c5.large entscheiden. Dies ist ein Typ mit hoher Verfügbarkeit, durchschnittlicher Größe und Funktionalität.

- Betriebssystem

Dieses Feld gibt das Betriebssystem an, auf dem Ihre Spieleserver laufen — entweder Linux oder Windows. Der Standardwert ist Linux.

Ausgehende Datenübertragung (DTO)

In diesem Abschnitt können Sie die Kosten für den Datenverkehr zwischen Ihren Spieleclients und den Spieleservern abschätzen. Datenübertragungsgebühren fallen nur für ausgehenden Datenverkehr an. Eingehende Datenübertragung ist kostenlos.

Erweitern Sie auf der [Amazon GameLift ServersSeite Konfigurieren](#) von AWS Pricing Calculator den Eintrag Ausgehende Datenübertragung (DTO), und füllen Sie dann die folgenden Felder aus:

- Typ der DTO-Schätzung

Je nachdem, wie Sie die Datenübertragung für Ihr Spiel verfolgen, können Sie DTO auf eine der beiden folgenden Arten schätzen.

- Pro Monat (in GB) — Wenn du den monatlichen Traffic für deine Spieleserver verfolgst, wähle diesen Typ.
- Pro Spieler — Wenn du die Datenübertragung pro Spieler nachverfolgst, wähle diesen Typ. Dies ist der Standardtyp.

Im folgenden Feld schätzen Sie das DTO pro Spieler auf der Grundlage der Anzahl der Spielerstunden, die Sie im vorherigen Abschnitt berechnet haben.

- DTO pro Monat (in GB)

Wenn Sie den DTO-Schätztyp Pro Monat (in GB) ausgewählt haben, geben Sie Ihre geschätzte monatliche DTO-Nutzung in GB für jede Instanz pro Region ein.

- DTO pro Spieler

Wenn du die DTO-Schätzung pro Spieler ausgewählt hast, gib die geschätzte DTO-Nutzung deines Spiels pro Spieler in KB/s ein. Der Standardwert ist **4**.

Wenn du mit der Konfiguration deiner Amazon GameLift Servers Preisschätzung fertig bist, wähle **Zu meiner Schätzung hinzufügen**. Weitere Informationen zum Erstellen und Verwalten von Kostenvoranschlägen finden Sie im [AWS Pricing Calculator](#) [AWS Pricing Calculator Benutzerhandbuch unter Kostenvoranschlag erstellen, Service konfigurieren und weitere Dienste hinzufügen](#).

Strategien zur Kostenoptimierung

Verwenden Sie diese Strategien mit einer verwalteten Hosting-Lösung, um Ihre Cloud-Hosting-Kosten zu senken und gleichzeitig die hohe Leistung und das Spielerlebnis aufrechtzuerhalten.

Best Practices

Befolge diese Tipps, um Methoden zur Kostenoptimierung in dein Spiele-Hosting zu integrieren. Wir empfehlen dir, mindestens regelmäßige Überprüfungen deiner Amazon GameLift Servers Kosten und Nutzungsmuster zu vereinbaren. Wenn Sie Fragen zur Preisgestaltung haben, wenden Sie sich an Ihren

Flottenkonfiguration

- **Nutze den Preisrechner** — Verwende dieses Tool zusammen mit deinen Game-Hosting-Daten, um Optionen zu erkunden und mögliche Konfigurationsszenarien zur Kosteneinsparung zu testen. Siehe [AWS Preisrechner für Amazon GameLift Servers](#).
- **Sparen Sie bei der Entwicklung** — Verwenden Sie für Entwicklung und Tests kleinere, kostengünstigere Instance-Typen und führen Sie sie nur dann aus, wenn Sie sie benötigen.
- **Instance-Typ an Spiel anpassen** — Wählen Sie EC2 Amazon-Instance-Familien, Typen und Größen für Ihre Flotten, die den Anforderungen Ihres Spiels am besten entsprechen. Weitere Informationen finden Sie im Abschnitt Instanztypen unter [Wählen Sie Rechenressourcen für eine verwaltete Flotte](#).
- **Verwenden Sie Graviton-Instance-Typen** — Graviton-Instances werden von ARM-basierten Prozessoren angetrieben. Sie bieten ein besseres Preis-Leistungs-Verhältnis, eine höhere

Energieeffizienz und niedrigere Kosten als vergleichbare x86-basierte Instances mit dem On-Demand-Preismodell.

- Verwenden Sie Spot-Instances — Spot-Instances sind in der Regel kostengünstiger als On-Demand-Instances, aber möglicherweise nicht immer verfügbar. Kombinieren Sie Spot- und On-Demand-Flotten, um ein Gleichgewicht zwischen niedrigen Kosten und hoher Verfügbarkeit zu erzielen (eine Spot-Nutzung von 50-85% wird empfohlen). Verwenden Sie Spot-Instances für nicht kritische Spielmodi oder außerhalb der Spitzenzeiten. Um die Rentabilität von Spot-Instances zu überwachen, verfolgen Sie Metriken wie `InstanceInterruptions` und `GameServerInterruptions`. Siehe [On-Demand-Instances im Vergleich zu Spot Instances](#).
- Server unter Linux ausführen — Erstelle deine Gameserver-Runtime für Linux. Instances, die mit Linux bereitgestellt werden, sind in der Regel kostengünstiger als solche mit Windows.
- Optimieren Sie die Ressourcennutzung — Konfigurieren Sie Flotten für maximale Effizienz, indem Sie bei gleichbleibender Leistung so viele Spielserversprozesse wie möglich gleichzeitig ausführen. Siehe [Optimiere die Laufzeitkonfiguration des Spieleservers auf verwalteten Amazon GameLift Servers](#).
- Entwickeln Sie eine Strategie für mehrere Regionen — Die Bereitstellung von Game-Hosting an mehreren Standorten kann Ihren Spielern eine geringere Latenz und bessere Backup-Optionen bieten. Sorgen Sie für ein ausgewogenes Verhältnis zwischen regionaler Abdeckung und Kosteneffizienz, indem Sie das Hosting zunächst auf den Märkten Ihrer größten Anbieter bereitstellen und in Spitzenzeiten sekundäre Standorte für Überkapazitäten nutzen. Siehe [Erstellen Sie eine Warteschlange mit mehreren Standorten](#).

Skalierung der Flottenkapazität

- Auf Null skalieren — Wenn Flotten nicht genutzt werden, setzen Sie die Flottenkapazität manuell auf Null, um unnötige Gebühren zu vermeiden. Siehe [Manuelles Einstellen der Kapazität für eine Amazon GameLift Servers Flotte](#).
- Automatische Skalierung hinzufügen — Vermeiden Sie eine übermäßige Bereitstellung von Hosting-Ressourcen, indem Sie die Hosting-Kapazität für Spiele mithilfe von Auto Scaling anpassen. Passen Sie die Kapazität an die schwankende Spielernachfrage und andere wichtige Kennzahlen an. Siehe [Automatische Skalierung der Flottenkapazität mit Amazon GameLift Servers](#).
- Halten Sie einen Puffer bereit — Um plötzliche Spitzen der Spielernachfrage zu bewältigen, ohne die Spieler warten zu lassen, verwenden Sie Target Tracking, um einen Puffer inaktiver

Spielerserver aufrechtzuerhalten. Passen Sie den Puffer an die Größe und das Nutzungsverhalten Ihrer Spielerbasis an. Siehe [Zielbasiertes Auto Scaling](#).

Platzierung der Spielsitzungen

- Platzierungswarteschlangen verwenden — Amazon GameLift Servers Warteschlangen basieren auf einem Algorithmus, der anhand der Kosten für die Hosting-Ressourcen und anderer Faktoren die „bestmöglichen“ Austragungsorte für Spielsitzungen bestimmt. Siehe [Platzierung der Spielsitzungen konfigurieren](#).
- Passen Sie die Prioritäten der Warteschlangen an — Sie können ändern, wie eine Warteschlange die Hosting-Kosten bei der Platzierung von Spielsitzungen priorisiert. Siehe [Priorisieren Sie die Platzierung von Spielsitzungen](#).
- Überwachen Sie die Platzierungsmetriken — Verfolgen Sie, wie Warteschlangen Standorte für Spielsitzungen auswählen, um Optimierungsmöglichkeiten zu identifizieren. Siehe [Amazon GameLift Servers-Metriken für Warteschlangen](#).

Datenübertragung

- Datenübertragungen verfolgen — Überwachen Sie, wie Daten zwischen Ihren Spielclients und Servern übertragen werden, und ergreifen Sie Maßnahmen zur Optimierung der Aktivitäten.
- Datenkomprimierung verwenden — Erwägen Sie die Implementierung von Datenkomprimierungstechniken für Datenübertragungen zwischen Spielclients und Servern. Die Komprimierung spielt eine entscheidende Rolle bei der Reduzierung der Bandbreitennutzung sowie bei der Verbesserung der Spielleistung und Latenz.

Tools für Ressourcenkosten und -nutzung

Erfahren Sie, wie Sie mithilfe von AWS Tools Ihre Hosting-Kosten für Spiele überwachen und optimieren können Amazon GameLift Servers. Informationen zu zusätzlichen Tools finden Sie unter [AWS Fakturierung und Kostenmanagement](#).

AWS Tools für das Kostenmanagement

- Abrechnungskonsolle — Überprüfe deine AWS Rechnungen und deine Nutzung. Weitere Informationen [findest du unter Erste Schritte mit der Abrechnung](#).

- Benachrichtigungen zur Nutzung des kostenlosen Kontingents — Richten Sie Benachrichtigungen ein, wenn sich die Limits des kostenlosen Kontingents nähern. Weitere Informationen finden Sie unter [Nachverfolgung Ihrer Nutzung des AWS kostenlosen Kontingents](#).
- CloudWatch Amazon-Abrechnungsbenachrichtigungen — Konfigurieren Sie Benachrichtigungen, wenn die Nutzung benutzerdefinierte Schwellenwerte erreicht. Weitere Informationen [zur Übersicht Ihrer geschätzten AWS Gebühren finden Sie unter Einen Abrechnungsalarm erstellen](#).

Kostenzuweisung und -verfolgung

- Tags zur Kostenzuweisung — Tagge deine Flotten und andere Ressourcen, um die Hosting-Kosten zu organisieren und nachzuverfolgen. Weitere Informationen findest du unter [Kosten mithilfe von Tags zur AWS Kostenzuweisung organisieren und nachverfolgen](#).
- Kostenberichte — Erstellen Sie Berichte, die nach zugewiesenen Tags kategorisiert sind. Weitere Informationen finden Sie [unter Verwenden des monatlichen Kostenverteilungsberichts](#).
- AWS Cost Explorer— Analysieren Sie Kosten, Trends und Prognosen mithilfe von Filtern und anpassbaren Ansichten. Weitere Informationen finden Sie unter [Analysieren Ihrer Kosten und Nutzung mit AWS Cost Explorer](#).
- AWS Budgets— Verfolgen Sie Ihre AWS Kosten und Ihren Verbrauch und ergreifen Sie entsprechende Maßnahmen. Weitere Informationen finden Sie unter [Verwaltung Ihrer Kosten mit AWS Budgets](#).

Leistungsüberwachung mit Amazon GameLift Servers und Amazon CloudWatch

Überwachen Sie diese wichtigen Kennzahlen, um die Ressourcennutzung zu optimieren. Sehen Sie sich Metriken in der Amazon GameLift Servers Konsole an oder verwenden Sie CloudWatch Amazon-Dashboards. Einzelheiten zu allen verfügbaren Metriken für finden Sie Amazon GameLift Servers unter [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#).

- Instanzmetriken —ActiveInstances,IdleInstances, PercentIdleInstances
- Metriken für Serverprozesse —ActiveServerProcesses, HealthyServerProcesses
- Metriken für Spielsitzungen —ActiveGameSessions, AvailableGameSessions
- Metriken der Spielersitzungen — CurrentPlayerSessions
- Metriken für die Warteschlange —AverageWaitTime, QueueDepth

- Matchmaking-Metriken —CurrentTickets, MatchesPlaced
- Hardwareleistung
—CPUUtilization, NetworkIn/NetworkOut, DiskReadBytes/DiskWriteBytes
- Metriken zur Ressourcenauslastung
 - Wird verwendet PercentIdleInstances, um die optimale Flottengröße zu bestimmen.
 - Verfolge PercentAvailableGameSessions, um eine ausreichende Kapazität für das Hosten von Spielen sicherzustellen.
 - Überwachung InstanceInterruptions und Feststellung GameServerInterruptions der Rentabilität der Spot-Instance.

Amazon GameLift ServersReferenzhandbücher

Dieser Abschnitt enthält die Referenzdokumentation zur Verwendung von Amazon GameLift Servers.

Themen

- [Service-API für Amazon GameLift Servers](#)
- [Server-SDK 5.x für Amazon GameLift Servers](#)
- [Server-SDK für Amazon GameLift Servers Version 4 und früher](#)
- [Platzierungsveranstaltungen für Spielsitzungen](#)
- [Amazon GameLift ServersAMI-Versionen](#)
- [Amazon GameLift ServersServicestandorte](#)
- [Endpunkte und Kontingente von Amazon GameLift Servers](#)
- [API-Limits für Amazon GameLift Servers](#)
- [API-Beschränkungen für Amazon GameLift Servers SDK](#)
- [Amazon GameLift ServersUDP-Ping-Beacons](#)

Service-API für Amazon GameLift Servers

Verwenden Sie diese aufgabenbasierte Liste, um API-Operationen zu finden, wenn Sie Ihre Amazon GameLift Servers Game-Hosting-Lösungen und andere Funktionen erstellen. Das AWS SDK beinhaltet diese Operationen im `aws.gamelift` Namespace. [Laden Sie das AWS SDK](#) herunter oder [sehen Sie sich die Amazon GameLift Servers API-Referenzdokumentation](#) an. Sie können die API auch mit der AWS Befehlszeilenschnittstelle (AWS CLI) verwenden, wie in der [AWS CLI Befehlsreferenz](#) dokumentiert.

Die API umfasst zwei Gruppen von Vorgängen für das verwaltete Spiele-Hosting:

- [Amazon GameLift ServersHosting-Ressourcen verwalten](#)
- [Starte Spielsitzungen und trete Spielern bei](#)

Die Amazon GameLift Servers Service-API enthält auch Operationen zur Verwendung mit anderen Amazon GameLift Servers Tools und Lösungen. Eine Liste von FleetIQ APIs finden Sie unter [FleetIQAPI-Operationen](#). Eine Liste von Vorgängen FlexMatch APIs für das Matchmaking finden Sie unter [FlexMatchAPI-Operationen](#).

Amazon GameLift ServersHosting-Ressourcen verwalten

Rufen Sie diese Operationen auf, um Hosting-Ressourcen für Ihre Spieleserver zu konfigurieren, die Kapazität zu skalieren, um der Nachfrage der Spieler gerecht zu werden, Leistungs- und Nutzungsdaten zu erhalten und vieles mehr. Verwenden Sie diese API-Operationen beim Hosten von Spieleservern mit Amazon GameLift Servers, einschließlich Amazon GameLift ServersRealtime. Sie können für die meisten Aufgaben zur Ressourcenverwaltung auch in der [Amazon GameLift ServersKonsole](#) arbeiten oder Aufrufe mit dem Tool AWS Command Line Interface (AWS CLI) tätigen.

Bereiten Sie die Spieleserver für die Bereitstellung vor

Laden Sie den Gameservercode Ihres Spiels hoch und konfigurieren Sie ihn, um die Bereitstellung und den Start auf Hosting-Ressourcen vorzubereiten.

Verwalte benutzerdefinierte Gameserver-Builds

- [Upload-Build](#) — Laden Sie Build-Dateien von einem lokalen Pfad hoch und erstellen Sie eine neue Amazon GameLift Servers Build-Ressource. Dieser Vorgang, der als AWS CLI Befehl verfügbar ist, ist die gängigste Methode zum Hochladen von Spielservers-Builds.
- [CreateBuild](#) — Erstellen Sie einen neuen Build mit Dateien, die in einem Amazon S3 S3-Bucket gespeichert sind.
- [ListBuilds](#) — Ruft eine Liste aller Builds ab, die in eine Amazon GameLift Servers Region hochgeladen wurden.
- [DescribeBuild](#) — Ruft Informationen ab, die mit einem Build verknüpft sind.
- [UpdateBuild](#) — Ändern Sie die Build-Metadaten, einschließlich des Build-Namens und der Build-Version.
- [DeleteBuild](#) — Entfernt einen Build von Amazon GameLift Servers.

Amazon GameLift ServersRealtimeKonfigurationsskripte verwalten

- [CreateScript](#) — Laden Sie JavaScript Dateien hoch und erstellen Sie eine neue Amazon GameLift Servers Skriptressource.
- [ListScripts](#) — Ruft eine Liste aller Realtime Skripte ab, die in eine Amazon GameLift Servers Region hochgeladen wurden.
- [DescribeScript](#) — Ruft Informationen ab, die mit einem Realtime Skript verknüpft sind.

- [UpdateScript](#)— Ändern Sie die Skript-Metadaten und laden Sie den überarbeiteten Skriptinhalt hoch.
- [DeleteScript](#)— Entferne ein Realtime Skript von Amazon GameLift Servers.

Richten Sie Rechenressourcen für das Hosting ein

Konfiguriere Hosting-Ressourcen und stelle sie mit deinem Gameserver-Build- oder Realtime Konfigurationsskript bereit.

Flotten erstellen und verwalten

- [CreateFleet](#)— Konfiguriere und implementiere eine neue Amazon GameLift Servers Flotte von Computerressourcen für den Betrieb deiner Spieleserver. Nach der Bereitstellung werden die Spieleserver automatisch so gestartet, wie sie konfiguriert sind und bereit sind, Spielsitzungen abzuhalten.
- [ListFleets](#)— Ruft eine Liste aller Flotten in einer Amazon GameLift Servers Region ab.
- [DeleteFleet](#)— Entferne eine Flotte, die keine Spieleserver mehr betreibt oder Spieler hostet.
- Standorte der Flotte anzeigen/aktualisieren.
 - [CreateFleetLocations](#)— Fügen Sie entfernte Standorte zu einer bestehenden Flotte hinzu, die mehrere Standorte unterstützt
 - [DescribeFleetLocationAttributes](#)— Rufen Sie eine Liste aller abgelegenen Standorte für eine Flotte ab und sehen Sie sich den aktuellen Status jedes Standorts an.
 - [DeleteFleetLocations](#)— Entferne entfernte Standorte aus einer Flotte, die mehrere Standorte unterstützt.
- Anzeigen/Aktualisieren von Flottenkonfigurationen.
 - [DescribeFleetAttributes/UpdateFleetAttributes](#)— Die Metadaten und Einstellungen einer Flotte zum Schutz von Spielsitzungen und zur Begrenzung der Ressourcenerstellung anzeigen oder ändern.
 - [DescribeFleetPortSettings/UpdateFleetPortSettings](#)— Zeigt die für eine Flotte zulässigen Eingangsberechtigungen (IP-Adresse und Portbereich) an oder ändert sie.
 - [DescribeRuntimeConfiguration/UpdateRuntimeConfiguration](#)— Zeigen Sie an, welche Serverprozesse (und wie viele) auf jeder Instanz in einer Flotte ausgeführt werden sollen, oder ändern Sie sie.

Flottenkapazität verwalten

- [Beschreiben EC2 InstanceLimits](#) — Ruft die maximal zulässige Anzahl von Instanzen für das aktuelle AWS Konto und die aktuelle Nutzungsstufe ab.
- [DescribeFleetCapacity](#) — Ruft die aktuellen Kapazitätseinstellungen für die Heimatregion einer Flotte ab.
- [DescribeFleetLocationCapacity](#) — Rufen Sie die aktuellen Kapazitätseinstellungen für jeden Standort einer Flotte mit mehreren Standorten ab.
- [UpdateFleetCapacity](#) — Passen Sie die Kapazitätseinstellungen für eine Flotte manuell an.
- Einrichten von :
 - [PutScalingPolicy](#) — Aktivieren Sie zielbasiertes Auto Scaling, erstellen Sie eine benutzerdefinierte Auto Scaling-Richtlinie oder aktualisieren Sie eine bestehende Richtlinie.
 - [DescribeScalingPolicies](#) — Rufen Sie eine bestehende Auto Scaling-Richtlinie ab.
 - [DeleteScalingPolicy](#) — Löschen Sie eine Auto-Scaling-Richtlinie und verhindern Sie, dass sie die Kapazität einer Flotte beeinträchtigt.
 - [StartFleetActions](#) — Starten Sie die Auto Scaling-Richtlinien einer Flotte neu.
 - [StopFleetActions](#) — Setzen Sie die Auto-Scaling-Richtlinien einer Flotte aus.

Überwachen Sie die Flottenaktivität.

- [DescribeFleetUtilization](#) — Ruft Statistiken über die Anzahl der Serverprozesse, Spielsitzungen und Spieler ab, die derzeit in einer Flotte aktiv sind.
- [DescribeFleetLocationUtilization](#) — Ruft Nutzungsstatistiken für jeden Standort in einer Flotte mit mehreren Standorten ab.
- [DescribeFleetEvents](#) — Zeigt protokollierte Ereignisse für eine Flotte während eines bestimmten Zeitraums an.
- [DescribeGameSessions](#) — Ruft Metadaten einer Spielsitzung ab, einschließlich der Laufzeit eines Spiels und der aktuellen Spielerzahl.

Richten Sie Warteschlangen für die Platzierung von Spielsitzungen ein

Richten Sie Warteschlangen für mehrere Flotten und Regionen ein, um Spielsitzungen mit den besten verfügbaren Hosting-Ressourcen für Kosten, Latenz und Ausfallsicherheit zu platzieren.

- [CreateGameSessionQueue](#) — Erstelle eine Warteschlange, die bei der Bearbeitung von Anfragen für Platzierungen von Spielsitzungen verwendet werden kann.

- [DescribeGameSessionQueues](#)— Ruft die in einer Region definierten Warteschlangen für Spielsitzungen ab. Amazon GameLift Servers
- [UpdateGameSessionQueue](#)— Ändert die Konfiguration einer Warteschlange für Spielsitzungen.
- [DeleteGameSessionQueue](#)— Löscht eine Warteschlange für Spielsitzungen aus der Region.

Verwalten von Aliase

Verwenden Sie Aliase, um Ihre Flotten darzustellen oder ein zum Terminal alternatives Ziel zu erstellen. Aliase sind nützlich beim Übergang von Spielaktivitäten von einer Flotte zu einer anderen, wie z. B. bei Aktualisierungen des Spielserver-Builds.

- [CreateAlias](#)— Definiere einen neuen Alias und weise ihn optional einer Flotte zu.
- [ListAliases](#)— Ruft alle Flottenaliase ab, die in einer Amazon GameLift Servers Region definiert sind.
- [DescribeAlias](#)— Ruft Informationen zu einem vorhandenen Alias ab.
- [UpdateAlias](#)— Ändern Sie die Einstellungen für einen Alias, z. B. die Umleitung von einer Flotte zu einer anderen.
- [DeleteAlias](#)— Entferne einen Alias aus der Region.
- [ResolveAlias](#)— Ruft die Flotten-ID ab, auf die ein angegebener Alias verweist.

Connect zu verwalteten Hosting-Instanzen her

Zeigen Sie Informationen zu einzelnen Instances in einer Flotte an oder fordern Sie zur Fehlerbehebung Fernzugriff auf eine bestimmte Flotteninstance an.

- [DescribeInstances](#)— Rufen Sie Informationen zu jeder Instance in einer Flotte ab, einschließlich Instance-ID, IP-Adresse, Standort und Status.
- [GetInstanceAccess](#)— Fordern Sie die Zugangsdaten an, die für die Remoteverbindung zu einer bestimmten Instanz in einer Flotte erforderlich sind.

VPC-Peering einrichten

Erstellen und verwalten Sie VPC-Peering-Verbindungen zwischen Ihren Amazon GameLift Servers Hosting-Ressourcen und anderen AWS Ressourcen.

- [CreateVpcPeeringAuthorization](#)— Autorisieren Sie eine Peering-Verbindung zu einer Ihrer VPCs

- [DescribeVpcPeeringAuthorizations](#)— Rufen Sie gültige Autorisierungen für Peering-Verbindungen ab.
- [DeleteVpcPeeringAuthorization](#)— Löscht eine Autorisierung für eine Peering-Verbindung.
- [CreateVpcPeeringConnection](#)— Stellen Sie eine Peering-Verbindung zwischen der VPC für eine Amazon GameLift Servers Flotte und einer Ihrer Fahrzeuge her. VPCs
- [DescribeVpcPeeringConnections](#)— Rufen Sie Informationen zu aktiven oder ausstehenden VPC-Peering-Verbindungen mit einer Amazon GameLift Servers Flotte ab.
- [DeleteVpcPeeringConnection](#)— Löscht eine VPC-Peering-Verbindung mit einer Amazon GameLift Servers Flotte.

Starte Spielsitzungen und trete Spielern bei

Rufen Sie diese Operationen von einem Back-End-Dienst aus auf, um neue Spielsitzungen zu starten, Informationen über bestehende Spielsitzungen zu erhalten und Spieler zu Spielsitzungen hinzuzufügen. Diese Operationen sind für benutzerdefinierte Spieleserver vorgesehen, die auf Amazon GameLift Servers gehostet werden. Wenn Sie verwenden Amazon GameLift ServersRealtime, verwalten Sie Spielesitzungen mithilfe der [Amazon GameLift ServersRealtimeClient-API \(C#\)](#).

- Starten Sie neue Spielsitzungen für einen oder mehrere Spieler.
 - [StartGameSessionPlacement](#)— Fragen Sie Amazon GameLift Servers nach den besten verfügbaren Hosting-Ressourcen und starten Sie eine neue Spielsitzung. Dies ist die bevorzugte Methode zum Erstellen neuer Spielsitzungen. Es stützt sich auf Warteschlangen für Spielsitzungen, um die Verfügbarkeit von Hostings in mehreren Regionen zu verfolgen, und verwendet FleetIQ Algorithmen, um Platzierungen auf der Grundlage von Spielerlatenz, Hosting-Kosten, Standort usw. zu priorisieren.
 - [DescribeGameSessionPlacement](#)— Informiere dich über Einzelheiten und den Status einer Platzierungsanfrage.
 - [StopGameSessionPlacement](#)— Stornieren Sie eine Platzierungsanfrage.
 - [CreateGameSession](#)— Startet eine neue, leere Spielsitzung an einem bestimmten Flottenstandort. Dieser Vorgang gibt dir mehr Kontrolle darüber, wo die Spielsitzung beginnen soll, anstatt sie FleetIQ zur Bewertung der Platzierungsoptionen zu verwenden. Sie müssen der neuen Spielsitzung in einem separaten Schritt Spieler hinzufügen.
- Binden Sie Spieler in bestehende Spielsitzungen ein. Finde laufende Spielsitzungen mit verfügbaren Spielerplätzen und reserviere sie für neue Spieler.

- [CreatePlayerSession](#)— Reserviere einen offenen Slot für einen Spieler, um an einer Spielsitzung teilzunehmen.
- [CreatePlayerSessions](#)— Reservieren Sie offene Slots für mehrere Spieler, um an einer Spielsitzung teilzunehmen.
- Arbeiten Sie mit Spielsitzungs- und Spielersitzungsdaten. Verwalte Informationen zu Spielsitzungen und Spielersitzungen.
- [SearchGameSessions](#)— Fordere eine Liste der aktiven Spielsitzungen auf der Grundlage einer Reihe von Suchkriterien an.
- [DescribeGameSessions](#)— Rufen Sie Metadaten für bestimmte Spielsitzungen ab, einschließlich der aktiven Zeit und der aktuellen Spielerzahl.
- [DescribeGameSessionDetails](#)— Ruft Metadaten, einschließlich der Schutzeinstellungen für Spielsitzungen, für eine oder mehrere Spielsitzungen ab.
- [DescribePlayerSessions](#)— Informieren Sie sich über Spieleraktivitäten, einschließlich Status, Spielzeit und Spielerdaten.
- [UpdateGameSession](#)— Ändern Sie die Einstellungen für Spielsitzungen, z. B. die maximale Spielerzahl und die Beitrittsrichtlinien.
- [GetGameSessionLogUrl](#)— Ruft den Speicherort der gespeicherten Protokolle für eine Spielsitzung ab.

Server-SDK 5.x für Amazon GameLift Servers

Dieser Abschnitt enthält Referenzdokumentation für das Server-SDK 5.x für Amazon GameLift Servers. Das Server-SDK bietet Kernfunktionen, die dein Spieleserver für die Interaktion mit dem Amazon GameLift Servers Dienst verwendet. Ihr Spieleserver erhält beispielsweise vom Dienst Aufforderungen, Spielsitzungen zu starten und zu beenden, und er informiert den Dienst regelmäßig über den Status der Spielsitzungen. Integrieren Sie Ihre Spieleserver in das Server-SDK, bevor Sie sie zum Hosten einsetzen.

Verwenden Sie diese Server-SDK-Referenz, um Ihre benutzerdefinierten Multiplayer-Spieleserver zum Hosten zu integrieren Amazon GameLift Servers. Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Die neueste Hauptversion des Server-SDK für Amazon GameLift Servers ist 5.x. Die folgenden Hosting-Funktionen erfordern die Verwendung von Version 5.x:

- Amazon GameLift ServersIrgendwo

- Amazon GameLift ServersPlugin für Unreal Engine und Unity

Note

Wenn Sie das Server-SDK Version 4.x oder früher verwenden müssen, finden Sie [Server-SDK für Amazon GameLift Servers Version 4 und früher](#) die Dokumentation und Download-Informationen unter.

Themen

- [Updates im Server-SDK 5 für Amazon GameLift Servers](#)
- [Migrieren Sie zum Server-SDK 5.x für Amazon GameLift Servers](#)
- [C++-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
- [Go-Server-SDK für Amazon GameLift Servers -- Aktionen](#)
- [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)

Updates im Server-SDK 5 für Amazon GameLift Servers

Ihre gehosteten Spieleserver verwenden das Server-SDK für Amazon GameLift Servers die Kommunikation mit dem Amazon GameLift Servers Dienst, um Spielsitzungen für Spieler zu starten und zu verwalten. Die neueste Version, das Amazon GameLift Servers Server-SDK 5, bietet eine Reihe von Verbesserungen und Unterstützung für neue Amazon GameLift Servers Funktionen. Wenn Ihr Spieleserver-Build derzeit Amazon GameLift Servers Server-SDK 4 oder eine frühere Version verwendet, folgen Sie den Anweisungen in diesem Thema, um Ihre Spiele zu aktualisieren.

Amazon GameLift ServersServer-SDK Version 5.0.0 und höher beinhaltet die folgenden Updates:

- Erweiterte Sprachen — Bibliotheken sind in den folgenden Sprachen verfügbar: C++, C#, Go. Sie können die C++-Bibliotheken für die Verwendung mit Unreal Engine erstellen.
- Unterstützung für Game Engine-Plugins — Die Amazon GameLift Servers eigenständigen Plugins für Unreal Engine und Unity benötigen Amazon GameLift Servers Server-SDK 5-Bibliotheken. Diese Plugins bieten geführte Workflows zum Integrieren, Testen und Bereitstellen deiner Spiele Amazon GameLift Servers zum Hosten. Siehe [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#) und [Amazon GameLift ServersPlugin für Unreal Engine](#) Dokumentation.

- **Amazon GameLift Servers Anywhere-Support** — Mit Anywhere-Flotten können Sie Ihre eigenen Hosting-Ressourcen einrichten, um Amazon GameLift Servers Funktionen (einschließlich Matchmaking) zu nutzen. Fügen Sie den Amazon GameLift Servers Agenten hinzu, um die Lebenszyklusverwaltung von Spielsitzungen zu automatisieren. Verwenden Sie Anywhere-Flotten für das Produktionshosting mit Hardware vor Ort oder richten Sie Testumgebungen für eine schnelle iterative Spieleentwicklung ein. [Siehe Überall, Hosting und der Amazon GameLift Servers Agent.](#)
- **Aktualisierte Testtools** — Mit der Amazon GameLift Servers Anywhere-Funktion können Sie lokale oder cloudbasierte Testumgebungen für Ihre Spiele einrichten. Richten Sie Tests mit oder ohne den Amazon GameLift Servers Agenten ein. Diese Tools ersetzen Amazon GameLift Servers Local. Siehe [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten.](#)
- **Konsolidierte .NET-Lösung für C#** — Das C#-Server-SDK 5.1+ unterstützt .NET Framework 4.6.2 (aktualisiert von 4.6.1) und .NET 6.0 in einer einzigen Lösung. .NET Standard 2.1 ist mit den von Unity erstellten Bibliotheken verfügbar.
- **Neue Compute Ressource** — Diese neue Ressource kombiniert verschiedene Arten von Hosting-Ressourcen. Sie umfasst cloudbasierte Hosting-Ressourcen (verwaltete EC2 oder Container-Flotten) und kundengesteuerte Hosting-Ressourcen (Anywhere-Flotten). Es beinhaltet die folgenden Updates:
 - Zu den neuen API-Aufrufen für die Compute Ressource gehören:
[ListComputeDescribeCompute\(\)](#), [\(\)](#) und [GetComputeAccess\(\)](#). Diese Aktionen geben Hosting-Ressourceninformationen für jede Art von Amazon GameLift Servers Flotte zurück. Im Allgemeinen sollten Sie bei Flotten mit Spieleservern, die das Server-SDK 5.x verwenden, die rechenspezifischen Aktionen verwenden, um instanzspezifische Aktionen zu ersetzen. [Darüber hinaus sind diese Aktionen für die Verwendung in Anywhere-Flotten ohne den Amazon GameLift Servers Agenten vorgesehen: RegisterCompute\(\), DeregisterCompute \(\) und \(\).](#)
[GetComputeAuthToken](#)
 - Neue Metrik `ActiveCompute` mit den CloudWatch Dimensionen `FleetIdLocation`, und `ComputeType`. Diese Metrik ersetzt die vorherige Metrik `ActiveInstances`.
- **Amazon EC2 Systems Manager (SSM) für Fernzugriff** — Verwenden Sie für zusätzliche Sicherheit SSM anstelle von SSH, wenn Sie eine Verbindung zu Instances in Amazon GameLift Servers verwalteten Flotten herstellen. Siehe [Stellen Sie eine Remoteverbindung zu Amazon GameLift Servers Flotteninstanzen her.](#)

Migrieren Sie zum Server-SDK 5.x für Amazon GameLift Servers

Um ein Spieleprojekt so zu aktualisieren, dass es das Server-SDK Version 5.x verwendet, nehmen Sie die folgenden Änderungen vor:

Note

Wenn du das Server-SDK Version 4.x oder früher weiterhin verwenden musst, findest du die Dokumentation und Download-Informationen [Server-SDK für Amazon GameLift Servers Version 4 und früher](#) unter. Speziell für die SDK-Version 4.0.2 können Sie sie von den [offiziellen GitHub](#) Versionen herunterladen.

1. Holen Sie sich das neueste Server-SDK für Amazon GameLift Servers das Paket für Ihre Entwicklungsumgebung [[Download-Seite](#)]. Folgen Sie den Installationsanweisungen in der Readme Datei für Ihr heruntergeladenes Paket und Ihre heruntergeladene Version. In dieser Anleitung erfährst du, wie du den Server SDKs mit deinem Spielprojekt verwendest.
 - [Für Entwicklungsumgebungen, die C++, C# oder Go verwenden](#)
 - [Für Unreal Engine-Projekte \(nur C++-Server-SDK für Unreal-Bibliotheken\)](#)
 - [Für Unity-Projekte \(C#-Server-SDK nur für Unity-Bibliotheken\)](#)
 - [Zur Verwendung mit dem Amazon GameLift Servers Plugin für Unreal Engine](#)
 - [Zur Verwendung mit dem Amazon GameLift Servers Plugin für Unity](#)
2. Aktualisieren Sie Ihren Servercode wie folgt:
 - Ändern Sie die Servercode-Callback-Funktion `onCreateGameSession()` auf `onStartGameSession()`.
 - Aktualisieren Sie die `InitSDK()` Eingaben nach Bedarf:
 - Wenn Sie planen, den Spielserver-Build entweder auf einer Amazon GameLift Servers verwalteten EC2-Flotte oder auf einer Anywhere-Flotte mit dem Amazon GameLift Servers Agenten bereitzustellen, gehen Sie wie folgt vor:

Aufruf `InitSDK()` ohne Parameter ([C++](#)) ([C#](#)) ([Unreal](#)). Dieser Aufruf richtet die Rechenumgebung und eine WebSocket Verbindung zum Amazon GameLift Servers Dienst ein.
 - Wenn du vorhast, den Gameserver-Build auf einer Anywhere-Flotte ohne den Amazon GameLift Servers Agenten bereitzustellen:

Rufen Sie `InitSDK()` mit Serverparametern ([C++](#)) ([C#](#)) ([Unreal](#)) auf. Ein Spieleserver-Prozess verwendet diese Parameter, um eine Verbindung mit dem Amazon GameLift Servers Dienst herzustellen.

3. Wenn dein Gameserver-Build oder andere gehostete Anwendungen während der Ausführung mit anderen AWS Ressourcen kommunizieren, musst du ändern, wie die Anwendung auf diese Ressourcen zugreift. Ersetzen Sie die Verwendung von `AssumeRoleCredentials()` durch die neue Server-SDK-Aktion `GetFleetRoleCredentials()` (für Spieleserver) oder verwenden Sie gemeinsam genutzte Anmeldeinformationen (für andere Anwendungen). Weitere Informationen zur Implementierung dieser Änderung finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).
4. Wenn Ihr Projekt die Server-SDK-Aktion `GetInstanceCertificate()` zum Abrufen eines TLS-Zertifikats aufgerufen hat, ändern Sie Ihren Code so, dass er stattdessen das neue `GetComputeCertificate()` ([C++](#)) ([C#](#)) ([Unreal](#)) verwendet.
5. Wenn du deinen Spiel-Build hochlädst Amazon GameLift Servers (z. B. mit [upload-build](#) oder [CreateBuild\(\)](#)), setze den `ServerSdkVersion` Parameter auf die 5.x-Version, die du verwendest (dieser Parameter ist derzeit standardmäßig auf 4.0.2 voreingestellt). Dieser Parameter muss mit den tatsächlichen Server-SDK-Bibliotheken im Spieleserver-Build übereinstimmen. Wenn Sie die falsche Version für einen hochgeladenen Spieleserver-Build angeben, schlagen alle Flotten fehl, die Sie mit diesem Build erstellen. Siehe [Erstelle einen Gameserver-Build für Amazon GameLift Servers](#).

Das folgende Beispiel zeigt, wie die Server-SDK-Version angegeben wird:

```
aws gamelift upload-build \  
  --operating-system AMAZON_LINUX_2023 \  
  --server-sdk-version "5.0.0" \  
  --build-root "~/mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-west-2
```

6. Wenn Sie Skripts verwenden, um eine Remoteverbindung zu verwalteten Flotten herzustellen, aktualisieren Sie die Skripts so, dass sie den neuen Prozess verwenden, wie unter [beschrieben Stellen Sie eine Remoteverbindung zu Amazon GameLift Servers Flotteninstanzen her](#).

C++-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen

Verwenden Sie die Server-SDK 5.x-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrieren. Amazon GameLift Servers Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Note

In diesem Thema wird die Amazon GameLift Servers C++-API beschrieben, die Sie verwenden können, wenn Sie mit der C++-Standardbibliothek (std) bauen. Insbesondere bezieht sich diese Dokumentation auf Code, den Sie mit der `-DDGAMELIFT_USE_STD=1` Option kompilieren.

C++-Server-SDK 5.x für Amazon GameLift Servers -- Datentypen

Verwenden Sie die Amazon GameLift Servers C++-Server-SDK 5.x-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrieren. Amazon GameLift Servers Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Note

In diesem Thema wird die Amazon GameLift Servers C++-API beschrieben, die Sie verwenden können, wenn Sie mit der C++-Standardbibliothek (std) bauen. Insbesondere bezieht sich diese Dokumentation auf Code, den Sie mit der `-DDGAMELIFT_USE_STD=1` Option kompilieren.

[C++-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)

Datentypen

- [LogParameters](#)
- [MetricsParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)

- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [AttributeValue](#)
- [GetFleetRoleCredentialsRequest](#)
- [AwsLongOutcome](#)
- [AwsStringOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [GenericOutcome](#)
- [GenericOutcomeCallable](#)
- [PlayerSession](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [Initialisieren SDKOutcome](#)
- [GameLiftError](#)
- [Aufzählungen](#)

LogParameters

Ein Objekt, das Dateien identifiziert, die während einer Spielsitzung generiert wurden und die Sie nach dem Ende der Spielsitzung hochladen und speichern möchten Amazon GameLift Servers. Der Spieleserver stellt LogParameters es Amazon GameLift Servers als Teil eines ProcessParameters Objekts in einem [ProcessReady\(\)](#) Aufruf bereit.

Eigenschaften	Beschreibung

LogPaths	Die Liste der Verzeichnispfade zu den Spielserver-Protokolldateien, die Sie für den future Amazon GameLift Servers Zugriff speichern möchten. Der Serverprozess generiert diese Dateien während jeder Spielsitzung. Du definierst Dateipfade und -namen auf deinem Spielserver und speicherst sie im Hauptverzeichnis des Spiels. Die Protokollpfade müssen absolut sein. Wenn Ihr Game-Build beispielsweise Spielsitzungsprotokolle in einem Pfad wie speichert <code>MyGame\sessionLogs\</code>, dann wäre der Pfad <code>c:\game\MyGame\sessionLogs</code> auf einer Windows-Instanz. Typ: <code>std::vector<std::string></code> Required: No

MetricsParameters

Verwenden Sie diesen Datentyp, um die Erfassung von Messwerten und die Absturzberichterstattung für den Spielserver zu konfigurieren. Der Spielserver kommuniziert `MetricsParameters` mit Amazon GameLift Servers in einem [InitMetrics\(\)](#) Anruf.

Eigenschaften	Beschreibung
StatsdHost	Der Hostname oder die IP-Adresse des StatsD-Servers für die Erfassung von Metriken. Typ: <code>std::string</code> Required: No
StatsdPort	Die Portnummer des StatsD-Servers für die Erfassung von Metriken.

	Typ: <code>int</code> Required: No
CrashReporterHost	Der Hostname oder die IP-Adresse des Crash-Reporter-Servers. Typ: <code>std::string</code> Required: No
CrashReporterPort	Die Portnummer des Crash-Reporter-Servers. Typ: <code>int</code> Required: No
FlushIntervalMs	Das Intervall in Millisekunden für die Übertragung von Metrikdaten an den Server. Typ: <code>int</code> Required: No
MaxPacketSize	Die maximale Größe in Byte für Messpakete, die an den Server gesendet werden. Typ: <code>int</code> Required: No

ProcessParameters

Dieser Datentyp enthält den Satz von Parametern, an den Amazon GameLift Servers in a gesendet wird [ProcessReady\(\)](#).

Eigenschaften	Beschreibung
LogParameters	Ein Objekt mit Verzeichnispfaden zu Dateien, die während einer Spielsitzung generiert

	werden. Amazon GameLift Servers kopiert und speichert die Dateien für den future Zugriff. Typ: <code>Aws::GameLift::Server::</code> LogParameters Required: No
OnHealthCheck	Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift Servers ruft diese Funktion alle 60 Sekunden auf und wartet 60 Sekunden auf eine Antwort. Der Serverprozess kehrt zurück, TRUE wenn er fehlerfrei ist, FALSE wenn nicht. Wenn keine Antwort zurückgegeben wird, wird der Amazon GameLift Servers Serverprozess als fehlerhaft aufgezeichnet. Typ: <code>std::function<bool()></code> <code>onHealthCheck</code> Required: No
OnProcessTerminate	Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um das Herunterfahren des Serverprozesses zu erzwingen. Amazon GameLift Servers wartet nach dem Aufruf dieser Funktion 5 Minuten, bis der Serverprozess heruntergefahren ist, und antwortet mit einem ProcessEnding() Aufruf, bevor er den Serverprozess herunterfährt. Typ: <code>std::function<void()></code> <code>onProcessTerminate</code> Erforderlich: Ja

OnStartGameSession

Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um eine neue Spielsitzung zu aktivieren. Amazon GameLift Servers ruft diese Funktion als Antwort auf eine Client-Anfrage auf. [CreateGameSession](#) Die Callback-Funktion übergibt ein [GameSession](#) Objekt.

```
Typ: const std::function<void  
(Aws::GameLift::Model::Game  
Session)> onStartGameSession
```

Erforderlich: Ja

OnUpdateGameSession

Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift Servers ruft diese Funktion auf, wenn eine Match-Backfill-Anfrage bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Sie übergibt ein [GameSession](#) Objekt, ein Status-Update (updateReason) und die Match-Backfill-Ticket-ID.

```
Typ: std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)> onUpdateGameSession
```

Required: No

Port

Die Portnummer, auf der der Serverprozess auf neue Spielerverbindungen wartet. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: Integer

Erforderlich: Ja

UpdateGameSession

Dieser Datentyp aktualisiert ein Spielsitzungsobjekt, das den Grund für die Aktualisierung der Spielsitzung und die zugehörige Backfill-Ticket-ID enthält, falls Backfill zum Füllen von Spielersitzungen in der Spielsitzung verwendet wird.

Eigenschaften	Beschreibung
GameSession	Ein GameSession-Objekt. Das GameSession Objekt enthält Eigenschaften, die eine Spielsitzung beschreiben. Typ: <code>Aws::GameLift::Server::GameSession</code> Erforderlich: Ja
UpdateReason	Der Grund, warum die Spielsitzung aktualisiert wird. Typ: <code>Aws::GameLift::Server::UpdateReason</code> Erforderlich: Ja

Eigenschaften	Beschreibung
BackfillTicketId	Die ID des Backfill-Tickets, das versucht, die Spielsitzung zu aktualisieren. Typ: <code>std::string</code> Required: No

GameSession

Dieser Datentyp enthält Details einer Spielsitzung.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Eine Spielsitzung ARN hat das folgende Format: <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Typ: <code>std::string</code> Required: No
Name	Eine beschreibende Bezeichnung der Spielsitzung. Typ: <code>std::string</code> Required: No
FleetId	Eine eindeutige Kennung für die Flotte, auf der die Spielsitzung läuft. Typ: <code>std::string</code> Required: No

Eigenschaften	Beschreibung
MaximumPlayerSessionCount	Die maximale Anzahl von Spielerverbindungen zur Spielsitzung. Typ: <code>int</code> Required: No
Port	Die Portnummer für die Spielsitzung. Um eine Verbindung zu einem Amazon GameLift Servers Spieleserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: <code>int</code> Required: No
IpAddress	Die IP-Adresse der Spielsitzung. Um eine Verbindung zu einem Amazon GameLift Servers Spieleserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: <code>std::string</code> Required: No
GameSessionData	Eine Reihe von benutzerdefinierten Spielsitzungseigenschaften, die als einzelner Zeichenfolgenwert formatiert sind. Typ: <code>std::string</code> Required: No

Eigenschaften	Beschreibung
MatchmakerData	Informationen über den Matchmaking-Prozess, der zur Erstellung der Spielsitzung verwendet wurde, in JSON-Syntax, formatiert als Zeichenfolge. Zusätzlich zur verwendeten Matchmaking-Konfiguration enthält es Daten zu allen Spielern, die dem Spiel zugewiesen wurden, einschließlich Spielerattributen und Teamzuweisungen. Typ: <code>std::string</code> Required: No
GameProperties	Eine Reihe von benutzerdefinierten Eigenschaften für eine Spielsitzung, formatiert als Schlüssel/Wert-Paare. Diese Eigenschaften werden zusammen mit der Aufforderung, eine neue Spielsitzung zu starten, übergeben. Typ: <code>std :: vector < GameProperty ></code> Required: No

Eigenschaften	Beschreibung
DnsName	Die DNS-ID, die der Instanz zugewiesen ist, auf der die Spielsitzung ausgeführt wird. Werte haben das folgende Format: • TLS-fähige Flotten: <code><unique identifier>.<region identifier>.amazon gamelift.com</code> • Non-TLS-enabled Flotten: <code>ec2-<unique identifier>.compute.amazonaws.com</code> Wenn du eine Verbindung zu einer Spielsitzung herstellst, die auf einer TLS-fähigen Flotte läuft, musst du den DNS-Namen verwenden, nicht die IP-Adresse. Typ: <code>std::string</code> Required: No

ServerParameters

Informationen, die ein Spieleserver-Prozess verwendet, um eine Verbindung mit dem Amazon GameLift Servers Dienst herzustellen. Geben Sie diese Parameter beim Aufrufen [InitSDK\(\)](#) nur an, wenn der Spieleserver-Build für eine Anywhere-Flotte oder eine Containerflotte ohne den Amazon GameLift Servers Agenten bereitgestellt wird. Rufen Sie für alle anderen Einsatzszenarien [InitSDK\(\)](#) ohne Parameter auf.

Eigenschaften	Beschreibung
webSocketUrl	Das <code>GameLiftServerSdkEndpoint</code> Amazon GameLift Servers kehrt zurück, wenn Sie RegisterCompute nach einer Amazon GameLift Servers Anywhere-Rechenressource suchen.

Eigenschaften	Beschreibung
	Typ: <code>std::string</code> Erforderlich: Ja
Prozess-ID	Eine eindeutige Kennung, die für den Serverprozess registriert ist, der dein Spiel hostet. Typ: <code>std::string</code> Erforderlich: Ja
hostId	Das HostID ist die, die bei der Registrierung Ihres Computers <code>ComputeName</code> verwendet wurde. Weitere Informationen finden Sie unter RegisterCompute. Typ: <code>std::string</code> Erforderlich: Ja
FleetID	Die eindeutige Kennung der Flotte, für die der Computer registriert ist. Weitere Informationen finden Sie unter RegisterCompute. Typ: <code>std::string</code> Erforderlich: Ja
AuthToken	Das dadurch generierte Authentifizierungstoken authentifiziert Ihren Server gegenüber Amazon GameLift Servers. Weitere Informationen finden Sie unter GetComputeAuthToken. Typ: <code>std::string</code> Erforderlich: Ja

StartMatchBackfillRequest

Informationen, die zur Erstellung einer Matchmaking-Backfill-Anfrage verwendet wurden. Der Spieleserver übermittelt diese Informationen Amazon GameLift Servers in einem Anruf an.

[StartMatchBackfill\(\)](#)

Eigenschaften	Beschreibung
GameSessionArn	Eine eindeutige Kennung für die Spielsitzung. Die API-Operation GetGameSessionId gibt den Bezeichner im ARN-Format zurück. Typ: <code>std::string</code> Erforderlich: Ja
MatchmakingConfigurationArn	Eine eindeutige Kennung in Form eines ARN, die der Matchmaker für diese Anfrage verwenden kann. Der Matchmaker-ARN für die ursprüngliche Spielsitzung befindet sich im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Weitere Informationen zu Matchmaker-Daten finden Sie unter Arbeiten mit Matchmaker-Daten. Typ: <code>std::string</code> Erforderlich: Ja
Players	Ein Datensatz, der alle Spieler repräsentiert, die an der Spielsitzung teilnehmen. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Typ: <code>std::vector<Player></code> Erforderlich: Ja

Eigenschaften	Beschreibung
TicketId	Eine eindeutige Kennung für ein Matchmaking- oder Match-Backfill-Anfrageticket. Amazon GameLift ServersGeneriert einen, wenn Sie keinen Wert angeben. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzuberechnen. Typ: <code>std::string</code> Required: No

Player

Dieser Datentyp steht für einen Spieler beim Matchmaking. Wenn ein Spieler eine Matchmaking-Anfrage startet, verfügt er über eine Spieler-ID, Attribute und möglicherweise Latenzdaten. Amazon GameLift Serversfügt Teaminformationen hinzu, nachdem ein Spiel gespielt wurde.

Eigenschaften	Beschreibung
LatencyInMS	Eine Reihe von Werten, ausgedrückt in Millisekunden, die die Latenz angeben, die ein Spieler erlebt, wenn er mit einem Standort verbunden ist. Wenn diese Eigenschaft verwendet wird, wird der Player nur für die aufgelisteten Standorte abgeglichen. Wenn ein Matchmaker eine Regel hat, die die Latenz der Spieler auswertet , müssen die Spieler die zu vergleichende Latenz melden. Typ: <code>Dictionary<string,int></code> Required: No

Eigenschaften	Beschreibung
PlayerAttributes	Eine Sammlung von Schlüssel/Wert-Paaren, die Spielerinformationen zur Verwendung bei der Spielerzuweisung enthalten. Die Schlüssel der Spielerattribute müssen mit den in einem Matchmaking-Regelsatz PlayerAttributes verwendeten übereinstimmen. Weitere Informationen zu Spielerattributen finden Sie unter AttributeValue. Typ: <code>std::map<std::string, AttributeValue></code> Required: No
PlayerId	Eine eindeutige Kennung für einen Spieler. Typ: <code>std::string</code> Required: No
Team	Der Name der Mannschaft, der der Spieler in einem Spiel zugewiesen wird. Sie definieren den Teamnamen im Matchmaking-Regelsatz. Typ: <code>std::string</code> Required: No

DescribePlayerSessionsRequest

Ein Objekt, das angibt, welche Spilersitzungen abgerufen werden sollen. Der Serverprozess stellt diese Informationen mit einem [DescribePlayerSessions\(\)](#) Aufruf von bereitAmazon GameLift Servers.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielsitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code>. Das <code>GameSessionID</code> ist eine benutzerdefinierte ID-Zeichenfolge oder eine Typ: <code>std::string</code> Required: No
PlayerSessionId	Die eindeutige Kennung für eine Spielsitzung. Verwenden Sie diesen Parameter, um eine einzelne spezifische Spielsitzung anzufordern. Typ: <code>std::string</code> Required: No
PlayerId	Die eindeutige Kennung für einen Spieler. Verwenden Sie diesen Parameter, um alle Spielsitzungen für einen bestimmten Spieler anzufordern. Siehe Spieler generieren IDs. Typ: <code>std::string</code> Required: No
PlayerSessionStatusFilter	Der Sitzungsstatus des Spielers, nach dem die Ergebnisse gefiltert werden sollen. Zu den möglichen Sitzungsstatus eines Spielers gehören:

Eigenschaften	Beschreibung
	• RESERVIERT — Die Anfrage für die Spiellersitzung wurde empfangen, aber der Spieler hat keine Verbindung zum Serverprozess hergestellt oder wurde nicht validiert. • AKTIV — Der Spieler wurde durch den Serverprozess validiert und ist verbunden. • ABGESCHLOSSEN — Die Spielerverbindung wurde unterbrochen. • TIMEDOUT — Eine Anfrage für eine Spiellersitzung wurde empfangen, aber der Spieler hat keine Verbindung hergestellt oder wurde nicht innerhalb des Timeoutlimits (60 Sekunden) bestätigt. Typ: <code>std::string</code> Required: No
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite anzeigt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Spiellersitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: <code>std::string</code> Required: No
Limit	Die maximale Anzahl der auszugebenden Ergebnisse. Wenn Sie eine Spiellersitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: <code>int</code> Required: No

StopMatchBackfillRequest

Informationen, die verwendet werden, um eine Matchmaking-Backfill-Anfrage zu stornieren. Der Spieleserver übermittelt diese Informationen in einem Anruf an den Amazon GameLift Servers Service. [StopMatchBackfill\(\)](#)

Eigenschaften	Beschreibung
GameSessionArn	Eine eindeutige Spielsitzungs-ID der Anfrage, die storniert wird. Typ: <code>char[]</code> Required: No
MatchmakingConfigurationArn	Eine eindeutige Kennung des Matchmakers, an den diese Anfrage gesendet wurde. Typ: <code>char[]</code> Required: No
TicketId	Eine eindeutige Kennung des Tickets für die Backfill-Anfrage, das storniert werden soll. Typ: <code>char[]</code> Required: No

AttributeValue

Verwenden Sie diese Werte in [Player](#) Attribut-Schlüssel-Wert-Paaren. Mit diesem Objekt können Sie einen Attributwert mit einem der gültigen Datentypen angeben: Zeichenfolge, Zahl, Zeichenkettenarray oder Datenmap. Jedes `AttributeValue` Objekt muss genau eine der verfügbaren Eigenschaften verwenden: SN,SL, oderSDM.

Eigenschaften	Description
AttrType	Gibt den Typ des Attributwerts an. Zu den möglichen Attributwerttypen gehören:

Eigenschaften	Description
	• NONE • ZEICHENFOLGE • DOPPELT • STRING_LIST • STRING_DOUBLE_MAP Required: No
S	Stellt einen Zeichenketten-Attributwert dar. Typ: <code>std::string</code> Required: No
N	Stellt einen numerischen Attributwert dar. Typ: <code>double</code> Required: No
SL	Stellt ein Array von Zeichenkettenattributwerten dar. Typ: <code>std::vector<std::string></code> Required: No
SDM	Stellt ein Wörterbuch mit Zeichenkettenschlüsseln und Doppelwerten dar. Typ: <code>std::map<std::string, double></code> Required: No

GetFleetRoleCredentialsRequest

Dieser Datentyp gewährt dem Spieleserver eingeschränkten Zugriff auf deine anderen Ressourcen. AWS Weitere Informationen finden Sie unter [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#).

Eigenschaften	Beschreibung
RoleArn	Der Amazon-Ressourcenname (ARN) der Servicerolle, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen erweitert. Typ: <code>std::string</code> Required: No
RoleSessionName	Der Name der Rollensitzung, mit dem Sie eine AWS -Security-Token-Service AssumeRole Sitzung eindeutig identifizieren können. Dieser Name wird in Auditprotokollen wie denen in angezeigt CloudTrail. Typ: <code>std::string</code> Required: No

AwsLongOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: <code>long</code> Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann.

Eigenschaften	Description
	Typ: <code>long&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

AwsStringOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: <code>std::string</code> Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>long&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code>

Eigenschaften	Description
	Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

DescribePlayerSessionsOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “DescribePlayerSessionsResult” Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>Aws::GameLift::Server::Model::DescribePlayerSessionsResult&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError”

Eigenschaften	Description
	Required: No

DescribePlayerSessionsResult

Eine Sammlung von Objekten mit Eigenschaften für jede Spilersitzung, die der Anfrage entspricht.

Eigenschaften	Description
NextToken	Ein Token, das den Beginn der nächsten sequentiellen Ergebnisseite angibt. Verwenden Sie das Token, das bei einem vorherigen Aufruf dieses Vorgangs zurückgegeben wurde. Um am Anfang der Ergebnismenge zu beginnen, geben Sie keinen Wert an. Wenn eine Spilersitzungs-ID angegeben ist, wird dieser Parameter ignoriert. Typ: <code>std::string</code> Erforderlich: Ja
PlayerSessions	Typ: <code>IList<the section called "PlayerSession"></code> Erforderlich:
ResultWithOwnership	Das Ergebnis der Aktion wird in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>std::string&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja

Eigenschaften	Description
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GenericOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GenericOutcomeCallable

Dieser Datentyp ist ein asynchrones generisches Ergebnis. Er besitzt die folgenden Eigenschaften:

Eigenschaften	Description
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja

Eigenschaften	Description
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

PlayerSession

Dieser Datentyp steht für eine Spilersitzung, die an den Spieleserver weitergegeben wird. Amazon GameLift Servers Weitere Informationen finden Sie unter [PlayerSession](#).

Eigenschaften	Description
CreationTime	Typ: long Required: No
FleetId	Typ: std::string Required: No
GameSessionId	Typ: std::string Required: No
IpAddress	Typ: std::string Required: No
PlayerData	Typ: std::string Required: No
PlayerId	Typ: std::string Required: No
PlayerSessionId	Typ: std::string

Eigenschaften	Description
	Required: No
Port	Typ: <code>int</code> Required: No
Status	Status der Spielsitzung für die Filterung der Ergebnisse. Wenn ein <code>PlayerSessionId</code> oder angegeben <code>PlayerId</code> wird, <code>PlayerSessionStatusFilter</code> hat das keine Auswirkung auf die Antwort. Typ: Eine <code>PlayerSessionStatus</code> Aufzählung. Die folgenden Werte sind möglich: • <code>ACTIVE</code> • <code>COMPLETED</code> • <code>NOT_FESTGELEGT</code> • <code>RESERVIERT</code> • <code>TIMEOUT</code> Required: No
TerminationTime	Typ: <code>long</code> Required: No
DnsName	Typ: <code>std::string</code> Required: No

StartMatchBackfillOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion.

Eigenschaften	Description
	Typ: the section called “StartMatchBackfillResult” Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>StartMatchBackfillResult&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

StartMatchBackfillResult

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
TicketId	Eine eindeutige Kennung für ein Matchmaking-Ticket. Wenn hier keine Ticket-ID angegeben ist, Amazon GameLift Servers wird eine in Form einer UUID generiert. Verwenden Sie diese Kennung, um den Status des Match-Backfill-Tickets zu verfolgen und die Spielergebnisse abzurufen. Typ: <code>std::string</code>

Eigenschaften	Description
	Required: No

GetComputeCertificateOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “GetComputeCertificateResult” Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>Aws::GameLift::Server::Model::GetComputeCertificateResult&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GetComputeCertificateResult

Der Pfad zum TLS-Zertifikat auf Ihrem Computer und der Hostname des Computers.

Eigenschaften	Description
CertificatePath	Der Pfad zum TLS-Zertifikat auf Ihrer Rechenressource. Wenn Sie eine Amazon GameLift Servers verwaltete Flotte verwenden , enthält dieser Pfad: • <code>certificate.pem</code> : Das Endbenutzerzertifikat. Die vollständige Zertifikatskette ist die Kombination aus den an dieses Zertifikat <code>certificateChain.pem</code> angehängten Zertifikaten. • <code>certificateChain.pem</code> : Die Zertifikatskette, die das Stammzertifikat und die Zwischenzertifikate enthält. • <code>rootCertificate.pem</code> : Das Stammzertifikat. • <code>privateKey.pem</code> : Der private Schlüssel für das Endbenutzerzertifikat. Typ: <code>std::string</code> Required: No
ComputeName	Der Name Ihrer Rechenressource. Typ: <code>std::string</code> Required: No

GetFleetRoleCredentialsOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “GetFleetRoleCredentialsResult” Required: No

Eigenschaften	Description
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>Aws::GameLift::Server::Model::GetFleetRoleCredentialsResult</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

GetFleetRoleCredentialsResult

Eigenschaften	Description
AccessKeyId	Die Zugriffsschlüssel-ID zur Authentifizierung und Bereitstellung des Zugriffs auf Ihre Ressourcen. AWS Typ: <code>string</code> Required: No
AssumedRoleId	Die ID des Benutzers, zu dem die Servicerolle gehört. Typ: <code>string</code> Required: No

Eigenschaften	Description
AssumedRoleUserArn	Der Amazon-Ressourcenname (ARN) des Benutzers, zu dem die Servicerolle gehört. Typ: <code>string</code> Required: No
Ablauf	Die Zeitspanne, bis Ihre Sitzungsanmeldedaten ablaufen. Typ: <code>DateTime</code> Required: No
SecretAccessKey	Die geheime Zugriffsschlüssel-ID für die Authentifizierung. Typ: <code>string</code> Required: No
SessionToken	Ein Token zur Identifizierung der aktuellen aktiven Sitzung, die mit Ihren AWS Ressourcen interagiert. Typ: <code>string</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

Initialisieren SDKOutcome

Note

`InitSDKOutcome` wird nur zurückgegeben, wenn Sie das SDK mit dem Flag erstellen. `std` Wenn Sie mit dem `nostd` Flag bauen, [the section called “GenericOutcome”](#) wird es stattdessen zurückgegeben.

Eigenschaften	Description
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GameLiftError

Eigenschaften	Description
ErrorType	Der Fehlertyp. Typ: Eine Aufzählung. GameLiftErrorType Required: No
ErrorMessage	Der Name des Fehlers. Typ: <code>std::string</code> Required: No

Eigenschaften	Description
ErrorMessage	Die Fehlermeldung. Typ: <code>std::string</code> Required: No

Aufzählungen

Für das Server-SDK für Amazon GameLift Servers (C++) definierte Enums sind wie folgt definiert:

GameLiftErrorType

Zeichenkettenwert, der den Fehlertyp angibt. Gültige Werte sind:

- `BAD_REQUEST_EXCEPTION`
- `GAMESESSION_ID_NOT_SET` — Die Spielsitzungs-ID wurde nicht festgelegt.
- `INTERNAL_SERVICE_EXCEPTION`
- `LOCAL_CONNECTION_FAILED` — Die lokale Verbindung zu ist fehlgeschlagen. Amazon GameLift Servers
- `NETWORK_NOT_INITIALIZED` — Das Netzwerk wurde nicht initialisiert.
- `SERVICE_CALL_FAILED` — Ein Aufruf eines Dienstes ist fehlgeschlagen. AWS
- `WEBSOCKET_CONNECT_FAILURE`
- `WEBSOCKET_CONNECT_FAILURE_FORBIDDEN`
- `WEBSOCKET_CONNECT_FAILURE_UNGÜLTIGE_URL`
- `TIMEOUT FÜR WEBSOCKET_CONNECT_FAILURE_`
- `ALREADY_INITIALIZED` — Der Server oder Client wurde bereits mit `Initialize ()` initialisiert. Amazon GameLift Servers
- `FLEET_MISMATCH` — Die Zielflotte entspricht nicht der Flotte einer `GameSession` oder `PlayerSession`.
- `GAMELIFT_CLIENT_NOT_INITIALIZED` — Der Client wurde nicht initialisiert. Amazon GameLift Servers
- `GAMELIFT_SERVER_NOT_INITIALIZED` — Amazon GameLift Servers Der Server wurde nicht initialisiert.

- `GAME_SESSION_ENDED_FAILED` — Das Server-SDK für konnte den Dienst nicht kontaktieren, um zu melden, dass die Spielsitzung beendet wurde. Amazon GameLift Servers
- `GAME_SESSION_NOT_READY` — Die Server-Spielsitzung wurde nicht aktiviert. Amazon GameLift Servers
- `GAME_SESSION_READY_FAILED` — Das Server-SDK für konnte den Dienst nicht kontaktieren, um zu melden, dass die Spielsitzung bereit ist. Amazon GameLift Servers
- `INITIALIZATION_MISMATCH` — Eine Client-Methode wurde nach `Server: :Initialize ()` aufgerufen oder umgekehrt.
- `NOT_INITIALIZED` — Der Amazon GameLift Servers Server oder Client wurde nicht mit `Initialize ()` initialisiert.
- `NO_TARGET_ALIASID_SET` — Eine Ziel-AliasID wurde nicht gesetzt.
- `NO_TARGET_FLEET_SET` — Es wurde keine Zielflotte festgelegt.
- `PROCESS_ENDING_FAILED` — Das Server-SDK für Amazon GameLift Servers konnte den Dienst nicht kontaktieren, um zu melden, dass der Prozess beendet wird.
- `PROCESS_NOT_ACTIVE` — Der Serverprozess ist noch nicht aktiv, nicht an einen gebunden und kann ihn weder annehmen noch verarbeiten. `GameSession PlayerSessions`
- `PROCESS_NOT_READY` — Der Serverprozess ist noch nicht bereit, aktiviert zu werden.
- `PROCESS_READY_FAILED` — Das Server-SDK für Amazon GameLift Servers konnte den Dienst nicht kontaktieren, um zu melden, dass der Prozess bereit ist.
- `SDK_VERSION_DETECTION_FAILED` — Die SDK-Versionserkennung ist fehlgeschlagen.
- `STX_CALL_FAILED` — Ein Aufruf der Server-Backend-Komponente ist fehlgeschlagen. `XStx`
- `STX_INITIALIZATION_FAILED` — Die Server-Backend-Komponente konnte nicht initialisiert werden. `XStx`
- `UNEXPECTED_PLAYER_SESSION` — Der Server hat eine nicht registrierte Spielsitzung festgestellt.
- `WEBSOCKET_CONNECT_FAILURE`
- `WEBSOCKET_CONNECT_FAILURE_FORBIDDEN`
- `WEBSOCKET_CONNECT_FAILURE_UNGÜLTIGE_URL`
- `TIMEOUT FÜR WEBSOCKET_CONNECT_FAILURE_`
- `WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE` — Behebbarer Fehler beim Senden einer Nachricht an den Dienst. `GameLift WebSocket`

- `WEBSOCKET_SEND_MESSAGE_FAILURE` — Fehler beim Senden einer Nachricht an GameLift den WebSocket Dienst.
- `MATCH_BACKFILL_REQUEST_VALIDATION` — Die Validierung der Anfrage ist fehlgeschlagen.
- `PLAYER_SESSION_REQUEST_VALIDATION` — Die Validierung der Anfrage ist fehlgeschlagen.

PlayerSessionCreationPolicy

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert. Gültige Werte sind:

- `ACCEPT_ALL` – Akzeptiert alle neuen Spiellersitzungen.
- `DENY_ALL` – Verwehrt neue Spiellersitzungen.
- `NOT_SET` — Die Spielsitzung ist nicht so eingestellt, dass neue Spiellersitzungen akzeptiert oder verweigert werden.

C++-Server-SDK 5.x für Amazon GameLift Servers -- Datentypen

Themen

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)

- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Zerstören \(\)](#)

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
Aws::GameLift::AwsStringOutcome Server::GetSdkVersion();
```

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als [the section called “AwsStringOutcome”](#)-Objekt zurück. Das zurückgegebene Objekt enthält die Versionsnummer (Beispiel 5.0.0). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitMetrics()

Initialisiert das Metriksystem für die Erfassung und Berichterstattung von Serverleistungsdaten. Die besten Ergebnisse erzielen Sie, wenn Sie diese Methode zuvor aufrufen, [InitSDK\(\)](#) um die Erfassung von Metriken während der SDK-Initialisierung zu aktivieren.

Syntax

```
Aws::GameLift::GenericOutcome InitMetrics();  
Aws::GameLift::GenericOutcome InitMetrics(const  
    Aws::GameLift::Server::MetricsParameters &metricsParameters);
```


Parameters

[MetricsParameters](#) (optional)

Konfigurationsparameter für die Erfassung von Metriken. Falls nicht angegeben, werden Standardwerte verwendet, die durch Umgebungsvariablen überschrieben werden können.

Rückgabewert

Bei Erfolg wird ein [the section called “GenericOutcome”](#) Objekt zurückgegeben, das den Erfolg anzeigt. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
// Initialize with default parameters (uses environment variables if available)
// Defaults: localhost:8125 for StatsD, localhost:8126 for crash reporter
// FlushInterval: 10000ms, MaxPacketSize: 512 bytes
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::InitMetrics();
if (outcome.IsSuccess()) {
    // Metrics system initialized successfully
}

// Initialize with custom parameters
Aws::GameLift::Server::MetricsParameters metricsParams("localhost", 8125, "crash-host",
    8126, 5000, 1024);
Aws::GameLift::GenericOutcome customOutcome =
    Aws::GameLift::Server::InitMetrics(metricsParams);
if (customOutcome.IsSuccess()) {
    // Metrics system initialized with custom parameters
}
```

InitSDK()

Initialisiert das Amazon GameLift Servers-SDK. Rufen Sie diese Methode beim Start vor allen anderen Initialisierungsschritten auf, die sich auf Amazon GameLift Servers. Diese Aktion liest Serverparameter aus der Host-Umgebung, um die Kommunikation zwischen dem Spielserver-Prozess und dem Amazon GameLift Servers Dienst einzurichten. Sie verwendet ein Idempotenz-Token, sodass Sie diesen Aufruf sicher wiederholen können, wenn er fehlschlägt.

Wenn der Spielserver-Build ohne den Amazon GameLift Servers Agenten für eine Amazon GameLift Servers Anywhere-Flotte oder Containerflotte bereitgestellt wird, rufen Sie auf [InitSDK\(\)](#) und geben Sie eine Reihe von Serverparametern an.

Syntax

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK();
```

Rückgabewert

Gibt ein [the section called "Initialisieren SDKOutcome"](#) Objekt zurück, das angibt, ob der Serverprozess aufrufbereit ist. [ProcessReady\(\)](#)

Beispiel

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
    Aws::GameLift::Server::InitSDK();
```

InitSDK()

Initialisiert das Amazon GameLift Servers-SDK. Rufen Sie diese Methode beim Start vor allen anderen Initialisierungsschritten auf, die sich auf Amazon GameLift Servers. Diese Aktion erfordert eine Reihe von Serverparametern, um die Kommunikation zwischen dem Spielserver-Prozess und dem Amazon GameLift Servers Dienst einzurichten. Sie verwendet ein Idempotenz-Token, sodass Sie diesen Aufruf sicher wiederholen können, wenn er fehlschlägt.

Wenn der Spielserver-Build mit dem Amazon GameLift Servers Agenten auf einer Amazon GameLift Servers verwalteten EC2-Flotte oder auf einer Amazon GameLift Servers Anywhere-Flotte oder Containerflotte bereitgestellt werden soll, rufen Sie [InitSDK\(\)](#) ohne Serverparameter auf.

Syntax

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK(serverParameters);
```

Parameters

ServerParameters

Um einen Spieleserver auf einer Amazon GameLift Servers Anywhere-Flotte zu initialisieren, konstruieren Sie ein `ServerParameters` Objekt mit den folgenden Informationen:

- Die URL, mit der die WebSocket Verbindung zu Ihrem Spieleserver hergestellt wurde.
- Die ID des Prozesses, der zum Hosten deines Spieleservers verwendet wird.
- Die ID des Computers, auf dem deine Gameserver-Prozesse gehostet werden.
- Die ID der Amazon GameLift Servers Flotte, die Ihren Amazon GameLift Servers Anywhere-Computer enthält.
- Das durch den Amazon GameLift Servers Vorgang generierte Autorisierungstoken.

Rückgabewert

Gibt ein [the section called "Initialisieren SDKOutcome"](#) Objekt zurück, das angibt, ob der Serverprozess aufrufbereit ist. [ProcessReady\(\)](#)

Note

Wenn Aufrufe von für Spiele-Builds, die auf Anywhere-Flotten bereitgestellt werden, fehlschlagen, überprüfen Sie den `ServerSdkVersion` Parameter, der beim Erstellen der Build-Ressource verwendet wurde. `InitSDK()` Sie müssen diesen Wert explizit auf die verwendete Server-SDK-Version festlegen. Der Standardwert für diesen Parameter ist 4.x, was nicht kompatibel ist. Um dieses Problem zu beheben, erstellen Sie einen neuen Build und stellen Sie ihn in einer neuen Flotte bereit.

Beispiel

Amazon GameLift ServersIrgendwo, Beispiel

```
//Define the server parameters
std::string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
std::string processId = "PID1234";
std::string fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
std::string hostId = "HardwareAnywhere";
```

```
std::string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
Aws::GameLift::Server::Model::ServerParameters serverParameters =
    Aws::GameLift::Server::Model::ServerParameters(webSocketUrl, authToken, fleetId,
        hostId, processId);

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
    Aws::GameLift::Server::InitSDK(serverParameters);
```

ProcessReady()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess bereit ist, Spielsitzungen zu hosten. Rufen Sie diese Methode nach dem Aufrufen auf [InitSDK\(\)](#). Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

Parameters

processParameters

Ein [ProcessParameters](#)-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Namen der im Spielservercode implementierten Callback-Methoden, die der Amazon GameLift Servers Dienst aufruft, um mit dem Serverprozess zu kommunizieren.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu beliebigen spielsitzungsspezifischen Dateien, die Amazon GameLift Servers erfassen und speichern soll.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des [ProcessReady\(\)](#)-Aufrufs und der Delegate-Funktion.

```

// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");           // Example of a log file written by the
game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::onHealthCheck, this),
        std::bind(&Server::onUpdateGameSession, this),
        listenPort,
        Aws::GameLift::Server::LogParameters(logPaths)
    );

Aws::GameLift::GenericOutcome outcome =
    Aws::GameLift::Server::ProcessReady(processReadyParameter);

// Implement callback functions
void Server::onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome =
        Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void Server::onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool Server::onHealthCheck()
{
    bool health;
    // complete health evaluation within 60 seconds and set health
    return health;
}

```

ProcessReadyAsync()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess zum Hosten von Spielsitzungen bereit ist. Diese Methode sollte aufgerufen werden, nachdem der Serverprozess bereit ist, eine Spielsitzung zu veranstalten. Die Parameter geben die Namen der Callback-Funktionen Amazon GameLift Servers an, die unter bestimmten Umständen aufgerufen werden sollen. Diese Funktionen müssen im Spiel-Servercode implementiert sein.

Dieser Aufruf ist asynchroner. Wenn Sie einen synchronen Aufruf durchführen möchten, verwenden Sie [ProcessReady\(\)](#). Weitere Details finden Sie unter [Initialisieren Sie den Serverprozess](#).

Syntax

```
GenericOutcomeCallable ProcessReadyAsync(  
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameters

processParameters

Ein [ProcessParameters](#)-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Namen der im Spielservercode implementierten Callback-Methoden, die der Amazon GameLift Servers Dienst aufruft, um mit dem Serverprozess zu kommunizieren.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu beliebigen spielsitzungsspezifischen Dateien, die Amazon GameLift Servers erfassen und speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
// Set parameters and call ProcessReady  
std::string serverLog("serverOut.log");           // This is an example of a log file  
    written by the game server  
std::vector<std::string> logPaths;
```

```

logPaths.push_back(serverLog);
int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(std::bind(&Server::onStartGameSession, this,
        std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this), std::bind(&Server::OnHealthCheck,
            this),
        std::bind(&Server::OnUpdateGameSession, this), listenPort,
        Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcomeCallable outcome =
    Aws::GameLift::Server::ProcessReadyAsync(processReadyParameter);

// Implement callback functions
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool onHealthCheck()
{
    // perform health evaluation and complete within 60 seconds
    return health;
}

```

ProcessEnding()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess beendet wird. Ruft diese Methode nach allen anderen Bereinigungsaufgaben (einschließlich des Herunterfahrens der aktiven Spielsitzung) und vor dem Beenden des Vorgangs auf. Je nach Ergebnis von wird der `ProcessEnding()` Prozess mit Erfolg (0) oder Fehler (-1) beendet und generiert ein Flottenereignis. Wenn der Prozess mit einem Fehler beendet wird, ist das generierte Flottenereignis `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Syntax

```
Aws::GameLift::GenericOutcome processEndingOutcome =  
    Aws::GameLift::Server::ProcessEnding();
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird `ProcessEnding()` der Serverprozess aufgerufen, `Destroy()` bevor er mit einem Exit-Code für Erfolg oder Fehler beendet wird.

```
Aws::GameLift::GenericOutcome processEndingOutcome =  
    Aws::GameLift::Server::ProcessEnding();  
Aws::GameLift::Server::Destroy();  
  
// Exit the process with success or failure  
if (processEndingOutcome.IsSuccess()) {  
    exit(0);  
}  
else {  
    cout << "ProcessEnding() failed. Error: " <<  
        processEndingOutcome.GetError().GetErrorMessage();  
    exit(-1);  
}
```

ActivateGameSession()

Meldet, Amazon GameLift Servers dass der Serverprozess eine Spielsitzung aktiviert hat und nun bereit ist, Spielerverbindungen zu empfangen. Diese Aktion sollte als Teil der `onStartGameSession()` Callback-Funktion aufgerufen werden, nachdem alle Spielsitzungen initialisiert wurden.

Syntax

```
Aws::GameLift::GenericOutcome activateGameSessionOutcome =  
    Aws::GameLift::Server::ActivateGameSession();
```


Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel zeigt, dass es als Teil der `onStartGameSession()` Delegate-Funktion `ActivateGameSession()` aufgerufen wird.

```
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession();
}
```

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spiellersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt.

Syntax

```
GenericOutcome
UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCreationPolicy
newPlayerSessionPolicy);
```

Parameters

playerCreationSessionRichtlinie

Typ: `PlayerSessionCreationPolicy` [Aufzählungswert](#).

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
Aws::GameLift::GenericOutcome outcome =  
    Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die vom aktiven Serverprozess gehostet wird.

Bei Prozessen im Leerlauf, die während einer Spielsitzung nicht aktiviert wurden, gibt der Aufruf a [the section called “GameLiftError”](#) zurück.

Syntax

```
AwsStringOutcome GetGameSessionId()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als [the section called “AwsStringOutcome”](#)-Objekt zurück. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Bei Prozessen im Leerlauf, die nicht während einer Spielsitzung aktiviert werden, gibt der Aufruf `Success = True` und `GameSessionId = zurück""`.

Beispiel

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess ergreift eine Aktion, nachdem er einen

`onProcessTerminate()` Rückruf von Amazon GameLift Servers erhalten hat. Amazon GameLift Servers ruft `onProcessTerminate()` aus den folgenden Gründen auf:

- Wenn der Serverprozess einen schlechten Zustand gemeldet hat oder nicht darauf reagiert hat Amazon GameLift Servers.
- Wenn die Instanz während eines Scale-Down-Ereignisses beendet wird.
- [Wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.](#)

Syntax

```
AwsDateTimeOutcome GetTerminationTime()
```

Rückgabewert

Bei Erfolg wird die Kündigungszeit als Objekt `AwsDateTimeOutcome` zurückgegeben. Der Wert ist die Endzeit, ausgedrückt in verstrichenen Ticks, die seitdem vergangen sind. `0001 00:00:00` Beispielsweise entspricht der Wert für Datum und Uhrzeit den `2020-09-13 12:26:40 -000Z` Ticks. `637355968000000000` Wenn keine Kündigungszeit verfügbar ist, wird eine Fehlermeldung zurückgegeben.

Wenn der Prozess keine erhalten hat `ProcessParameters`. `OnProcessTerminate()` Callback, eine Fehlermeldung wird zurückgegeben. Weitere Hinweise zum Herunterfahren eines Serverprozesses finden Sie unter. [Reagieren Sie auf eine Benachrichtigung zum Herunterfahren des Serverprozesses](#)

Beispiel

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

AcceptPlayerSession()

Benachrichtigt Amazon GameLift Servers, dass ein Spieler mit der angegebenen Spielsitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Bestätigung benötigt. Amazon GameLift Servers überprüft, ob die Player-Sitzungs-ID gültig ist. Nachdem die Spielsitzung bestätigt wurde, wird der Status des Spieler-Slots von RESERVIERT auf AKTIV Amazon GameLift Servers geändert.

Syntax

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Parameters

playerSessionId

Eindeutige ID Amazon GameLift Servers, die ausgestellt wird, wenn eine neue Spielsitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird eine Verbindungsanforderung behandelt, die das Überprüfen und Ablehnen einer ungültigen Spielsitzung beinhaltet. IDs

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
playerSessionId)
{
    Aws::GameLift::GenericOutcome connectOutcome =
    Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
    if(connectOutcome.IsSuccess())
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(connectOutcome.GetError().GetMessage());
    }
}
```

RemovePlayerSession()

Benachrichtigt Amazon GameLift Servers, dass ein Spieler die Verbindung zum Serverprozess getrennt hat. Amazon GameLift Servers ändert daraufhin den Player-Slot auf „Verfügbar“.

Syntax

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Parameters

playerSessionId

Eindeutige ID Amazon GameLift Servers, die ausgestellt wird, wenn eine neue Spielsitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

Ruft Spielsitzungsdaten ab, einschließlich Einstellungen, Sitzungsmetadaten und Spielerdaten. Verwenden Sie diese Methode, um Informationen zu folgenden Themen abzurufen:

- Eine Einzelspieler-Sitzung
- Alle Spielsitzungen in einer Spielsitzung
- Alle Spielsitzungen, die mit einer einzigen Spieler-ID verknüpft sind

Syntax

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest  
    describePlayerSessionsRequest)
```

Parameters

[DescribePlayerSessionsRequest](#)

Ein [the section called “DescribePlayerSessionsRequest”](#) Objekt, das beschreibt, welche Spielsitzungen abgerufen werden sollen.

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein [the section called "DescribePlayerSessionsOutcome"](#)-Objekt mit einer Menge von Spielsitzungsobjekten zurück, die den Anforderungsparametern entsprechen.

Beispiel

In diesem Beispiel werden alle Spielsitzungen abgefragt, die aktiv mit einer bestimmten Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen NextToken und ihn auf 10 setzen, werden die Sitzungsdatensätze der ersten 10 Spieler Amazon GameLift Servers zurückgegeben, die der Anfrage entsprechen.

```
// Set request parameters
Aws::GameLift::Server::Model::DescribePlayerSessionsRequest request;
request.SetPlayerSessionStatusFilter(Aws::GameLift::Server::Model::PlayerSessionStatusMapper::G
request.SetLimit(10);
request.SetGameSessionId("the game session ID");    // can use GetGameSessionId()

// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::DescribePlayerSessions(request);
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Weitere Informationen finden Sie unter [FlexMatchBackfill-Funktion](#).

Diese Aktion ist asynchron. Wenn neue Spieler gematcht werden, Amazon GameLift Servers liefert es mithilfe der Callback-Funktion aktualisierte Matchmaker-Daten. OnUpdateGameSession()

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst [StopMatchBackfill\(\)](#) auf, um die ursprüngliche Anforderung abubrechen.

Syntax

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
    startBackfillRequest);
```

Parameters

StartMatchBackfillRequest

Ein StartMatchBackfillRequest Objekt, das die folgenden Informationen übermittelt:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Information ist optional. Wenn keine ID angegeben wird, Amazon GameLift Servers wird eine generiert.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert ist in den Matchmaker-Daten der Spielsitzung enthalten.
- Die ID der Spielsitzung, die aufgefüllt werden soll.
- Die verfügbaren Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Rückgabewert

Gibt ein [the section called "StartMatchBackfillOutcome"](#) Objekt mit der ID des Match-Backfill-Tickets zurück oder schlägt fehl und es wird eine Fehlermeldung angezeigt.

Beispiel

```
// Build a backfill request
std::vector<Player> players;
Aws::GameLift::Server::Model::StartMatchBackfillRequest startBackfillRequest;
startBackfillRequest.SetTicketId("1111aaaa-22bb-33cc-44dd-5555eeee66ff"); // optional,
    autogenerated if not provided
startBackfillRequest.SetMatchmakingConfigurationArn("arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"); //from the game
    session matchmaker data
startBackfillRequest.SetGameSessionArn("the game session ARN");           // can use
    GetGameSessionId()
startBackfillRequest.SetPlayers(players);                                 // from the
    game session matchmaker data

// Send backfill request
Aws::GameLift::StartMatchBackfillOutcome backfillOutcome =
    Aws::GameLift::Server::StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void Server::OnUpdateGameSession(Aws::GameLift::Server::Model::GameSession gameSession,
    Aws::GameLift::Server::Model::UpdateReason updateReason, std::string backfillTicketId)
```

```
{  
    // handle status messages  
    // perform game-specific tasks to prep for newly matched players  
}
```

StopMatchBackfill()

Bricht eine aktive Match-Backfill-Anfrage ab. Weitere Informationen finden Sie unter [FlexMatchBackfill-Funktion](#).

Syntax

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameters

[StopMatchBackfillRequest](#)

Ein StopMatchBackfillRequest Objekt, das das zu stornierende Matchmaking-Ticket identifiziert:

- Die der Backfill-Anfrage zugewiesene Ticket-ID.
- Der Matchmaker, an den die Backfill-Anfrage gesendet wurde.
- Die Spielsitzung, die mit der Backfill-Anfrage verknüpft ist.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
// Set backfill stop request parameters  
  
Aws::GameLift::Server::Model::StopMatchBackfillRequest stopBackfillRequest;  
stopBackfillRequest.SetTicketId("1111aaaa-22bb-33cc-44dd-5555eeee66ff");  
stopBackfillRequest.SetGameSessionArn("the game session ARN"); // can use  
    GetGameSessionId()  
stopBackfillRequest.SetMatchmakingConfigurationArn("arn:aws:gamelift:us-  
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig");  
// from the game session matchmaker data
```



```
Aws::GameLift::GenericOutcome stopBackfillOutcome =  
    Aws::GameLift::Server::StopMatchBackfill(stopBackfillRequest);
```

GetComputeCertificate()

Ruft den Pfad zu dem TLS-Zertifikat ab, mit dem die Netzwerkverbindung zwischen Ihrer Amazon GameLift Servers Anywhere-Rechenressource und verschlüsselt wurde. Amazon GameLift Servers Sie können den Zertifikatspfad verwenden, wenn Sie Ihr Computergerät bei einer Amazon GameLift Servers Anywhere-Flotte registrieren. Weitere Informationen finden Sie unter [RegisterCompute](#).

Syntax

```
GetComputeCertificateOutcome Server::GetComputeCertificate()
```

Rückgabewert

Gibt eine [the section called “GetComputeCertificateOutcome”](#) zurück.

Beispiel

```
Aws::GameLift::GetComputeCertificateOutcome certificate =  
    Aws::GameLift::Server::GetComputeCertificate();
```

GetFleetRoleCredentials()

Ruft Anmeldeinformationen für IAM-Rollen ab, die Amazon GameLift Servers zur Interaktion mit anderen berechtigen. AWS-Services Weitere Informationen finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).

Syntax

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest  
    request);
```

Parameters

[GetFleetRoleCredentialsRequest](#)

Rückgabewert

Gibt ein [the section called “GetFleetRoleCredentialsOutcome”](#)-Objekt zurück.

Beispiel

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
    getFleetRoleCredentialsRequest;
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
    Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Dieses Beispiel zeigt, wie der optionale RoleSessionName Wert verwendet wird, um der Sitzung mit den Anmeldeinformationen zu Prüfungszwecken einen Namen zuzuweisen. Wenn Sie keinen Namen für die Rollensitzung angeben, wird der Standardwert *[host-id] [fleet-id]-* verwendet.

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
    getFleetRoleCredentialsRequest;
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction");
getFleetRoleCredentialsRequest.SetRoleSessionName("MyFleetRoleSession");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
    Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Zerstören ()

Befreit das Amazon GameLift Servers Spieleserver-SDK aus dem Speicher. Es hat sich bewährt, diese Methode nach ProcessEnding() und vor dem Beenden des Prozesses aufzurufen. Wenn Sie eine Anywhere-Flotte verwenden und Serverprozesse nicht nach jeder Spielsitzung beenden, rufen Sie auf Destroy() und starten Sie dann InitSDK() erneut, bevor Sie benachrichtigen, Amazon GameLift Servers dass der Prozess bereit ist, eine Spielsitzung mit zu hosten. ProcessReady()

Syntax

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

Parameters

Es gibt keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
Aws::GameLift::GenericOutcome processEndingOutcome =  
    Aws::GameLift::Server::ProcessEnding();  
Aws::GameLift::Server::Destroy();  
  
// Exit the process with success or failure  
if (processEndingOutcome.IsSuccess()) {  
    exit(0);  
}  
else {  
    cout << "ProcessEnding() failed. Error: " <<  
    processEndingOutcome.GetError().GetErrorMessage();  
    exit(-1);  
}
```

C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen

Verwenden Sie die Server-SDK 5.x-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrieren. Amazon GameLift Servers Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#). Wenn Sie das Amazon GameLift Servers Plugin für Unity verwenden, finden Sie auch weitere Informationen unter [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#).

C#-Server-SDK 5.x für Amazon GameLift Servers -- Datentypen

Verwenden Sie die Amazon GameLift Servers C#-Server-SDK 5.x-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrieren. Amazon GameLift Servers Hinweise zum Integrationsprozess finden Sie unter. [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#) Wenn Sie das Amazon GameLift Servers Plugin für Unity verwenden, finden Sie auch weitere Informationen unter [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#).

[C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)

Datentypen

- [LogParameters](#)

- [MetricsParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)
- [AttributeValue](#)
- [AwsStringOutcome](#)
- [GenericOutcome](#)
- [MetricsManagerOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [PlayerSession](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [AwsDateTimeOutcome](#)
- [GameLiftError](#)
- [Aufzählungen](#)

LogParameters

Verwenden Sie diesen Datentyp, um zu ermitteln, auf welche Dateien, die während einer Spielsitzung generiert wurden, der Spieleserver Amazon GameLift Servers nach dem Ende der Spielsitzung

hochladen soll. Der Spieleserver kommuniziert `LogParameters` to Amazon GameLift Servers in einem [ProcessReady\(\)](#) Anruf.

Eigenschaften	Beschreibung
LogPaths	Die Liste der Verzeichnispfade zu den Spieleserver-Protokolldateien, die Sie für den future Amazon GameLift Servers Zugriff speichern möchten. Der Serverprozess generiert diese Dateien während jeder Spielsitzung. Du definierst Dateipfade und -namen auf deinem Spieleserver und speicherst sie im Hauptverzeichnis des Spiels. Die Protokollpfade müssen absolut sein. Wenn Ihr Game-Build beispielsweise Spielsitzungsprotokolle in einem Pfad wie <code>MyGame\sessionLogs\</code> speichert, dann wäre der Pfad <code>c:\game\MyGame\sessionLogs</code> auf einer Windows-Instanz. Typ: <code>List<String></code> Required: No

MetricsParameters

Verwenden Sie diesen Datentyp, um die Erfassung von Metriken und die Absturzberichterstattung für den Spieleserver zu konfigurieren. Der Spieleserver kommuniziert `MetricsParameters` mit Amazon GameLift Servers in einem [InitMetrics\(\)](#) Anruf.

Eigenschaften	Beschreibung
StatsdHost	Der Hostname oder die IP-Adresse des StatsD-Servers für die Erfassung von Metriken. Typ: <code>String</code>

	Required: No
StatsdPort	Die Portnummer des StatsD-Servers für die Erfassung von Metriken. Typ: Integer Required: No
CrashReporterHost	Der Hostname oder die IP-Adresse des Crash Reporter-Servers. Typ: String Required: No
CrashReporterPort	Die Portnummer des Crash-Reporter-Servers. Typ: Integer Required: No
FlushIntervalMs	Das Intervall in Millisekunden für die Übertragung von Metrikdaten auf den Server. Typ: Integer Required: No
MaxPacketSize	Die maximale Größe in Byte für Messpakete, die an den Server gesendet werden. Typ: Integer Required: No

ProcessParameters

Dieser Datentyp enthält den Satz von Parametern, an die Amazon GameLift Servers bei einem [ProcessReady\(\)](#) Aufruf gesendet wird.

Eigenschaften	Beschreibung
LogParameters	Das Objekt mit einer Liste von Verzeichnispfaden zu den Protokolldateien der Spielsitzungen. Typ: <code>Aws::GameLift::Server::LogParameters</code> Erforderlich: Ja
OnHealthCheck	Der Name der Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift Servers ruft diese Funktion alle 60 Sekunden auf. Nach dem Aufruf Amazon GameLift Servers wartet diese Funktion 60 Sekunden auf eine Antwort. Wenn keine Antwort eingeht, wird der Amazon GameLift Servers Serverprozess als fehlerhaft aufgezeichnet. Typ: <code>void OnHealthCheckDelegate()</code> Erforderlich: Ja
OnProcessTerminate	Der Name der Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion wartet Amazon GameLift Servers fünf Minuten auf die Beendigung des Serverprozesses und antwortet mit einem ProcessEnding()-Aufruf, bevor es den Serverprozess beendet. Typ: <code>void OnProcessTerminateDelegate()</code>

	Erforderlich: Ja
OnStartGameSession	Der Name der Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um eine neue Spielsitzung zu aktivieren. Amazon GameLift Servers ruft diese Funktion als Antwort auf die Client-Anfrage auf. CreateGameSession Die Callback-Funktion nimmt ein GameSession Objekt entgegen. Typ: void OnStartGameSession Delegate(GameSession) Erforderlich: Ja
OnUpdateGameSession	Der Name der Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift Servers ruft diese Funktion auf, wenn eine Match-Backfill-Anfrage bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Sie übergibt ein GameSession Objekt, ein Status-Update (updateReason) und die Match-Backfill-Ticket-ID. Typ: void OnUpdateGameSessionDelegate () UpdateGameSession Required: No

Port

Die Portnummer, auf der der Serverprozess nach neuen Player-Verbindungen sucht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: Integer

Erforderlich: Ja

UpdateGameSession

Aktualisierte Informationen für ein Spielsitzungsobjekt, einschließlich des Grundes für die Aktualisierung der Spielsitzung. Wenn sich das Update auf eine Aktion zum Auffüllen eines Spiels bezieht, beinhaltet dieser Datentyp die ID des Backfill-Tickets.

Eigenschaften	Beschreibung
GameSession	Ein GameSession-Objekt. Das GameSession Objekt enthält Eigenschaften, die eine Spielsitzung beschreiben. Typ: GameSession GameSession() Erforderlich: Ja
UpdateReason	Der Grund, warum die Spielsitzung aktualisiert wird. Typ: UpdateReason UpdateReason() Erforderlich: Ja
BackfillTicketId	Die ID des Backfill-Tickets, das versucht, die Spielsitzung zu aktualisieren.

Eigenschaften	Beschreibung
	Typ: <code>String</code>
	Erforderlich: Ja

GameSession

Details einer Spielsitzung.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Eine Spielsitzung ARN hat das folgende Format:<code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Typ: <code>String</code> Required: No
Name	Eine beschreibende Bezeichnung der Spielsitzung. Typ: <code>String</code> Required: No
FleetId	Eine eindeutige Kennung für die Flotte, auf der die Spielsitzung läuft. Typ: <code>String</code> Required: No
MaximumPlayerSessionCount	Die maximale Anzahl von Spielerverbindungen zur Spielsitzung. Typ: <code>Integer</code>

Eigenschaften	Beschreibung
	Required: No
Port	Die Portnummer für die Spielsitzung. Um eine Verbindung zu einem Amazon GameLift Servers Spielseserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: <code>Integer</code> Required: No
IpAddress	Die IP-Adresse der Spielsitzung. Um eine Verbindung zu einem Amazon GameLift Servers Spielseserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: <code>String</code> Required: No
GameSessionData	Eine Reihe von benutzerdefinierten Spielsitzungseigenschaften, die als einzelner Zeichenfolgenwert formatiert sind. Typ: <code>String</code> Required: No

Eigenschaften	Beschreibung
MatchmakerData	Die Informationen über den Matchmaking-Prozess, der zur Erstellung der Spielsitzung verwendet wurde, in JSON-Syntax, formatiert als Zeichenfolge. Zusätzlich zur verwendeten Matchmaking-Konfiguration enthält sie Daten zu allen Spielern, die dem Spiel zugewiesen wurden, einschließlich Spielerattributen und Teamzuweisungen. Typ: <code>String</code> Required: No
GameProperties	Eine Reihe von benutzerdefinierten Eigenschaften für eine Spielsitzung, formatiert als Schlüssel/Wert-Paare. Diese Eigenschaften werden zusammen mit der Aufforderung, eine neue Spielsitzung zu starten, übergeben. Typ: <code>Dictionary<string, string></code> Required: No

Eigenschaften	Beschreibung
DnsName	Die DNS-ID, die der Instanz zugewiesen ist, auf der die Spielsitzung ausgeführt wird. Werte haben das folgende Format: • TLS-fähige Flotten: <code><unique identifier>.<region identifier>.amazon gamelift.com</code> • Non-TLS-enabled Flotten: <code>ec2-<unique identifier>.compute.amazonaws.com</code> Wenn du eine Verbindung zu einer Spielsitzung herstellst, die auf einer TLS-fähigen Flotte läuft, musst du den DNS-Namen verwenden, nicht die IP-Adresse. Typ: String Required: No

ServerParameters

Informationen, die zur Aufrechterhaltung der Verbindung zwischen einem Amazon GameLift Servers Anywhere-Server und dem Amazon GameLift Servers Dienst verwendet werden. Diese Informationen werden verwendet, wenn neue Serverprozesse mit gestartet [InitSDK\(\)](#) werden. Verwenden Sie für Server, die auf Amazon GameLift Servers verwalteten EC2-Instances gehostet werden, ein leeres Objekt.

Eigenschaften	Beschreibung
WebSocketUrl	Das wurde <code>GameLiftServerSdkEndpoint</code> zurückgegeben, <code>RegisterCompute</code> als Sie Teil von Amazon GameLift Servers Anywhere waren.

Eigenschaften	Beschreibung
	Typ: <code>String</code> Erforderlich: Ja
ProcessId	Eine eindeutige Kennung, die für den Serverprozess registriert ist, der dein Spiel hostet. Typ: <code>String</code> Erforderlich: Ja
HostId	Eine eindeutige Kennung für den Host bei den Serverprozessen, die dein Spiel hosten. Die <code>hostId</code> wird <code>ComputeName</code> verwendet, als Sie Ihren Computer registriert haben. Weitere Informationen finden Sie unter RegisterCompute Typ: <code>String</code> Erforderlich: Ja
FleetId	Die Flotten-ID der Flotte, für die der Computer registriert ist. Weitere Informationen finden Sie unter RegisterCompute . Typ: <code>String</code> Erforderlich: Ja
AuthToken	Das dadurch Amazon GameLift Servers generierte Authentifizierungstoken authentifiziert Ihren Amazon GameLift Servers Server. Weitere Informationen finden Sie unter GetComputeAuthToken . Typ: <code>String</code> Erforderlich: Ja

StartMatchBackfillRequest

Informationen, die zur Erstellung einer Matchmaking-Backfill-Anfrage verwendet wurden. Der Spieleserver übermittelt diese Informationen Amazon GameLift Servers in einem Anruf an.

[StartMatchBackfill\(\)](#)

Eigenschaften	Beschreibung
GameSessionArn	Die eindeutige Kennung der Spielsitzung. Die API-Operation GetGameSessionId gibt den Bezeichner im ARN-Format zurück. Typ: <code>String</code> Erforderlich: Ja
MatchmakingConfigurationArn	Die eindeutige Kennung in Form einer ARN, die der Matchmaker für diese Anfrage verwenden soll. Der Matchmaker-ARN für die ursprüngliche Spielsitzung befindet sich im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Weitere Informationen zu Matchmaker-Daten finden Sie unter Arbeiten mit Matchmaker-Daten. Typ: <code>String</code> Erforderlich: Ja
Players	Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Typ: <code>List<Player></code> Erforderlich: Ja

Eigenschaften	Beschreibung
TicketId	Die eindeutige Kennung für ein Matchmaking- oder Match-Backfill-Anfrageticket. Amazon GameLift ServersGeneriert einen, wenn Sie keinen Wert angeben. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzuberechnen. Typ: <code>String</code> Required: No

Player

Repräsentiert einen Spieler beim Matchmaking. Wenn eine Matchmaking-Anfrage gestartet wird, hat ein Spieler eine Spieler-ID, Attribute und möglicherweise Latenzdaten. Amazon GameLift Serversfügt Teaminformationen hinzu, nachdem ein Spiel gespielt wurde.

Eigenschaften	Beschreibung
LatencyInMS	Eine Reihe von Werten, ausgedrückt in Millisekunden, die die Latenz angeben, die ein Spieler erlebt, wenn er mit einem Standort verbunden ist. Wenn diese Eigenschaft verwendet wird, wird der Player nur für die aufgelisteten Standorte abgeglichen. Wenn ein Matchmaker eine Regel hat, die die Latenz der Spieler auswertet , müssen die Spieler die zu vergleichende Latenz melden. Typ: <code>Dictionary<string, int></code> Required: No

Eigenschaften	Beschreibung
PlayerAttributes	Eine Sammlung von Schlüssel/Wert-Paaren, die Spielerinformationen zur Verwendung bei der Spielerzuweisung enthalten. Die Schlüssel der Spielerattribute müssen mit den in einem Matchmaking-Regelsatz PlayerAttributes verwendeten übereinstimmen. Weitere Informationen zu Spielerattributen finden Sie unter AttributeValue. Typ: Dictionary<string, Attribute Value Required: No
PlayerId	Eine eindeutige Kennung für einen Spieler. Typ: String Required: No
Team	Der Name der Mannschaft, der der Spieler in einem Spiel zugewiesen wird. Sie definieren den Teamnamen im Matchmaking-Regelsatz. Typ: String Required: No

DescribePlayerSessionsRequest

Dieser Datentyp wird verwendet, um anzugeben, welche Spielsitzung(en) abgerufen werden sollen. Es kann auf verschiedene Arten verwendet werden: (1) PlayerSessionId um eine bestimmte Spielsitzung anzufordern; (2) um alle Spielsitzungen in der angegebenen Spielsitzung GameSessionId anzufordern; oder (3) um alle Spielsitzungen für den angegebenen Spieler anzufordern; oder (3) bieten a, PlayerId um alle Spielsitzungen für den angegebenen

Spieler anzufordern. Verwenden Sie für große Sammlungen von Spielsitzungen die Paginierungsparameter, um Ergebnisse als aufeinander folgende Seiten abzurufen.

Eigenschaften	Beschreibung
GameSessionId	Die eindeutige Kennung der Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielsitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist wie folgt: <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code>. Der Wert von <code><ID string></code> ist entweder eine benutzerdefinierte ID-Zeichenfolge (sofern bei der Erstellung der Spielsitzung angegeben) oder eine generierte Zeichenfolge. Typ: String Required: No
PlayerSessionId	Die eindeutige Kennung für eine Spielsitzung. Typ: String Required: No
PlayerId	Die eindeutige Kennung für einen Spieler. Siehe Spieler generieren IDs. Typ: String Required: No
PlayerSessionStatusFilter	Der Sitzungsstatus des Spielers, nach dem die Ergebnisse gefiltert werden sollen. Mögliche Spielsitzungsstatus sind u. a.: RESERVIERT — Die Anfrage für die Spielsitzung wurde empfangen, aber der

Eigenschaften	Beschreibung
	Spieler hat sich noch nicht mit dem Serverprozess verbunden, der bestätigt and/or wurde. • ACTIVE – Der Spieler wurden vom Serverprozess validiert und ist derzeit verbunden. • COMPLETED – Die Spielerverbindung wurde beendet. • TIMEDOUT — Eine Anfrage für eine Spielsitzung wurde empfangen, aber der Spieler hat keine Verbindung hergestellt und das Timeout-Limit (60 Sekunden) and/or wurde nicht bestätigt. Typ: <code>String</code> Required: No
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite anzeigt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Spielsitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: <code>String</code> Required: No
Limit	Die maximale Anzahl der auszugebenden Ergebnisse. Wenn Sie eine Spielsitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: <code>int</code> Required: No

StopMatchBackfillRequest

Informationen, die verwendet werden, um eine Matchmaking-Backfill-Anfrage zu stornieren. Der Spieleserver übermittelt diese Informationen in einem Anruf an den Amazon GameLift Servers Service. [StopMatchBackfill\(\)](#)

Eigenschaften	Beschreibung
GameSessionArn	Die eindeutige Spielsitzungs-ID der Anfrage, die storniert wurde. Typ: <code>string</code> Erforderlich: Ja
MatchmakingConfigurationArn	Die eindeutige Kennung des Matchmakers, an den diese Anfrage gesendet wurde. Typ: <code>string</code> Erforderlich: Ja
TicketId	Die eindeutige Kennung des Tickets für die Backfill-Anfrage, das storniert werden soll. Typ: <code>string</code> Erforderlich: Ja

GetFleetRoleCredentialsRequest

Dieser Datentyp gewährt dem Spieleserver eingeschränkten Zugriff auf deine anderen AWS Ressourcen. Weitere Informationen finden Sie unter [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#).

Eigenschaften	Beschreibung
RoleArn	Der Amazon-Ressourcenname (ARN) der Servicerolle, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen erweitert.

Eigenschaften	Beschreibung
	Typ: <code>string</code> Erforderlich: Ja
RoleSessionName	Der Name der Sitzung, die die Verwendung der Rollenanmeldedaten beschreibt. Typ: <code>string</code> Required: No

AttributeValue

Verwenden Sie diese Werte in Attribut-Schlüssel-Wert-Paaren [Player](#). Mit diesem Objekt können Sie einen Attributwert mit einem der gültigen Datentypen angeben: Zeichenfolge, Zahl, Zeichenkettenarray oder Datenmap. Jedes `AttributeValue` Objekt kann nur eine der verfügbaren Eigenschaften verwenden.

Eigenschaften	Description
AttrType	Gibt den Typ des Attributwerts an. Typ: Ein <code>AttrType</code> Aufzählungswert . Required: No
S	Stellt einen Zeichenkettenattributwert dar. Typ: <code>string</code> Erforderlich: Ja
N	Stellt einen numerischen Attributwert dar. Typ: <code>double</code> Erforderlich: Ja
SL	Stellt ein Array von Zeichenkettenattributwerten dar.

Eigenschaften	Description
	Typ: <code>string[]</code> Erforderlich: Ja
SDM	Stellt ein Wörterbuch mit Zeichenkettenschlüsseln und Doppelwerten dar. Typ: <code>Dictionary<string, double></code> Erforderlich: Ja

AwsStringOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: <code>string</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

GenericOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

MetricsManagerOutcome

Stellt das Ergebnis eines [InitMetrics\(\)](#) Anrufs dar. Enthält entweder eine MetricsManager Instanz bei Erfolg oder Fehlerinformationen bei einem Fehler.

Ergebnis	Die MetricsManager Instanz für die Erfassung und Berichterstattung von Metriken. Typ: <code>MetricsManager</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError"

Required: No

DescribePlayerSessionsOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called "DescribePlayerSessionsResult" Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

DescribePlayerSessionsResult

Eigenschaften	Description
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite angibt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Spielersitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: <code>string</code> Erforderlich: Ja

Eigenschaften	Description
PlayerSessions	Eine Sammlung von Objekten mit Eigenschaften für jede Spielsitzung, die der Anfrage entspricht. Typ: <code>ICollection<the section called "PlayerSession"></code> Erforderlich:
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

PlayerSession

Eigenschaften	Description
CreationTime	Typ: <code>long</code> Erforderlich: Ja
FleetId	Typ: <code>string</code> Erforderlich: Ja
GameSessionId	Typ: <code>string</code> Erforderlich: Ja
IpAddress	Typ: <code>string</code>

Eigenschaften	Description
	Erforderlich: Ja
PlayerData	Typ: <code>string</code> Erforderlich: Ja
PlayerId	Typ: <code>string</code> Erforderlich: Ja
PlayerSessionId	Typ: <code>string</code> Erforderlich: Ja
Port	Typ: <code>int</code> Erforderlich: Ja
Status	Typ: Eine <code>PlayerSessionStatus</code> Aufzählung . Erforderlich: Ja
TerminationTime	Typ: <code>long</code> Erforderlich: Ja
DnsName	Typ: <code>string</code> Erforderlich: Ja

StartMatchBackfillOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called "StartMatchBackfillResult"

Eigenschaften	Description
	Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

StartMatchBackfillResult

Eigenschaften	Description
TicketId	Typ: <code>string</code> Erforderlich: Ja

GetComputeCertificateOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called "GetComputeCertificateResult" Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code>

Eigenschaften	Description
	Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

GetComputeCertificateResult

Der Pfad zum TLS-Zertifikat auf Ihrem Computer und der Hostname des Computers.

Eigenschaften	Description
CertificatePath	Typ: <code>string</code> Erforderlich: Ja
ComputeName	Typ: <code>string</code> Erforderlich: Ja

GetFleetRoleCredentialsOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called "GetFleetRoleCredentialsResult" Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code>

Eigenschaften	Description
	Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

GetFleetRoleCredentialsResult

Eigenschaften	Description
AccessKeyId	Die Zugriffsschlüssel-ID zur Authentifizierung und Bereitstellung des Zugriffs auf Ihre Ressourcen. AWS Typ: <code>string</code> Required: No
AssumedRoleId	Die ID des Benutzers, zu dem die Servicerolle gehört. Typ: <code>string</code> Required: No
AssumedRoleUserArn	Der Amazon-Ressourcenname (ARN) des Benutzers, zu dem die Servicerolle gehört. Typ: <code>string</code> Required: No
Ablauf	Die Zeitspanne, bis Ihre Sitzungsanmeldedaten ablaufen. Typ: <code>DateTime</code> Required: No

Eigenschaften	Description
SecretAccessKey	Die geheime Zugriffsschlüssel-ID für die Authentifizierung. Typ: <code>string</code> Required: No
SessionToken	Ein Token zur Identifizierung der aktuellen aktiven Sitzung, die mit Ihren AWS Ressourcen interagiert. Typ: <code>string</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

AwsDateTimeOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: <code>DateTime</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht.

Eigenschaften	Description
	Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

GameLiftError

Eigenschaften	Description
ErrorType	Der Fehlertyp. Typ: Eine Aufzählung. GameLiftErrorType Required: No
ErrorMessage	Der Name des Fehlers. Typ: <code>string</code> Required: No
ErrorName	Die Fehlermeldung. Typ: <code>string</code> Required: No

Aufzählungen

Für das Server-SDK für Amazon GameLift Servers (C#) definierte Enums sind wie folgt definiert:

AttrType

- NONE
- ZEICHENFOLGE
- DOPPELT
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

Zeichenkettenwert, der den Fehlertyp angibt. Gültige Werte sind:

- SERVICE_CALL_FAILED — Ein Aufruf eines Dienstes ist AWS fehlgeschlagen.
- LOCAL_CONNECTION_FAILED — Die lokale Verbindung zu ist fehlgeschlagen. Amazon GameLift Servers
- NETWORK_NOT_INITIALIZED — Das Netzwerk wurde nicht initialisiert.
- GAMESESSION_ID_NOT_SET — Die Spielsitzungs-ID wurde nicht festgelegt.
- BAD_REQUEST_EXCEPTION
- INTERNALE_SERVICE_EXCEPTION
- ALREADY_INITIALIZED — Der Amazon GameLift Servers Server oder Client wurde bereits mit Initialize () initialisiert.
- FLEET_MISMATCH — Die Zielflotte entspricht nicht der Flotte einer GameSession oder PlayerSession.
- GAMELIFT_CLIENT_NOT_INITIALIZED — Der Client wurde nicht initialisiert. Amazon GameLift Servers
- GAMELIFT_SERVER_NOT_INITIALIZED — Amazon GameLift Servers Der Server wurde nicht initialisiert.
- GAME_SESSION_ENDED_FAILED — Das Server-SDK für konnte den Dienst nicht kontaktieren, um zu melden, dass die Spielsitzung beendet wurde. Amazon GameLift Servers
- GAME_SESSION_NOT_READY — Die Server-Spielsitzung wurde nicht aktiviert. Amazon GameLift Servers
- GAME_SESSION_READY_FAILED — Das Server-SDK für konnte den Dienst nicht kontaktieren, um zu melden, dass die Spielsitzung bereit ist. Amazon GameLift Servers
- INITIALIZATION_MISMATCH — Eine Client-Methode wurde nach Server: :Initialize () aufgerufen oder umgekehrt.

- NOT_INITIALIZED — Der Amazon GameLift Servers Server oder Client wurde nicht mit Initialize () initialisiert.
- NO_TARGET_ALIASID_SET — Eine Ziel-AliasID wurde nicht gesetzt.
- NO_TARGET_FLEET_SET — Es wurde keine Zielflotte festgelegt.
- PROCESS_ENDING_FAILED — Das Server-SDK für Amazon GameLift Servers konnte den Dienst nicht kontaktieren, um zu melden, dass der Prozess beendet wird.
- PROCESS_NOT_ACTIVE — Der Serverprozess ist noch nicht aktiv, nicht an einen gebunden und kann ihn weder annehmen noch verarbeiten. GameSession PlayerSessions
- PROCESS_NOT_READY — Der Serverprozess ist noch nicht bereit, aktiviert zu werden.
- PROCESS_READY_FAILED — Das Server-SDK für Amazon GameLift Servers konnte den Dienst nicht kontaktieren, um zu melden, dass der Prozess bereit ist.
- SDK_VERSION_DETECTION_FAILED — Die SDK-Versionserkennung ist fehlgeschlagen.
- STX_CALL_FAILED — Ein Aufruf der Server-Backend-Komponente ist fehlgeschlagen. XStx
- STX_INITIALIZATION_FAILED — Die Server-Backend-Komponente konnte nicht initialisiert werden. XStx
- UNEXPECTED_PLAYER_SESSION — Der Server hat eine nicht registrierte Spielsitzung festgestellt.
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_UNGÜLTIGE_URL
- TIMEOUT FÜR WEBSOCKET_CONNECT_FAILURE_
- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE — Behebbarer Fehler beim Senden einer Nachricht an den Dienst. GameLift WebSocket
- WEBSOCKET_SEND_MESSAGE_FAILURE — Fehler beim Senden einer Nachricht an GameLift den WebSocket Dienst.
- MATCH_BACKFILL_REQUEST_VALIDATION — Die Validierung der Anfrage ist fehlgeschlagen.
- PLAYER_SESSION_REQUEST_VALIDATION — Die Validierung der Anfrage ist fehlgeschlagen.

PlayerSessionCreationPolicy

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert. Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielsitzungen.
- DENY_ALL – Verwehrt neue Spielsitzungen.
- NOT_SET — Die Spielsitzung ist nicht so eingestellt, dass neue Spielsitzungen akzeptiert oder verweigert werden.

PlayerSessionStatus

- ACTIVE
- COMPLETED
- NOT_SET
- RESERVIERT
- TIMEOUT

C#-Server-SDK 5.x für Amazon GameLift Servers -- Datentypen

Themen

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)

- [GetFleetRoleCredentials\(\)](#)
- [Zerstören \(\)](#)

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
AwsStringOutcome GetSdkVersion();
```

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als [the section called “AwsStringOutcome”](#)-Objekt zurück. Die zurückgegebene Zeichenfolge enthält die Versionsnummer (Beispiel 5.0.0). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

InitMetrics()

Initialisiert das Metriksystem für die Erfassung und Berichterstattung von Serverleistungsdaten. Die besten Ergebnisse erzielen Sie, wenn Sie diese Methode zuvor aufrufen, [InitSDK\(\)](#) um die Erfassung von Metriken während der SDK-Initialisierung zu aktivieren.

Syntax

```
MetricsManagerOutcome InitMetrics();  
MetricsManagerOutcome InitMetrics(MetricsParameters metricsParameters);
```

Parameters

[MetricsParameters](#) (optional)

Konfigurationsparameter für die Erfassung von Metriken. Falls nicht angegeben, werden Standardwerte verwendet, die durch Umgebungsvariablen überschrieben werden können.

Rückgabewert

Bei Erfolg wird ein [the section called “MetricsManagerOutcome”](#) Objekt zurückgegeben, das die MetricsManager Instanz enthält. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
// Initialize with default parameters (uses environment variables if available)
var outcome = GameLiftServerAPI.InitMetrics();
if (outcome.Success) {
    var metricsManager = outcome.Result;
} else {
    Console.WriteLine("Failed to initialize metrics: " + outcome.Error.ErrorMessage);
}

// Initialize with custom parameters
var metricsParams = new MetricsParameters("localhost", 8125, "crash-host", 9999, 1000,
    1024);
var customOutcome = GameLiftServerAPI.InitMetrics(metricsParams);
if (customOutcome.Success) {
    var metricsManager = customOutcome.Result;
}
```

InitSDK()

Initialisiert das Amazon GameLift Servers SDK für eine verwaltete EC2-Flotte. Rufen Sie diese Methode beim Start auf, bevor eine weitere Initialisierung im Zusammenhang mit erfolgt. Amazon GameLift Servers Diese Methode liest Serverparameter aus der Host-Umgebung, um die Kommunikation zwischen dem Server und dem Amazon GameLift Servers Dienst einzurichten. Sie verwendet ein Idempotenz-Token, sodass Sie diesen Aufruf sicher wiederholen können, wenn er fehlschlägt.

Syntax

```
GenericOutcome InitSDK();
```

Rückgabewert

Bei Erfolg wird ein InitSdkOutcome Objekt zurückgegeben, das anzeigt, dass der Serverprozess aufrufbereit ist. [ProcessReady\(\)](#)

Beispiel

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK();
```

InitSDK()

Initialisiert das Amazon GameLift Servers SDK für eine Anywhere-Flotte. Rufen Sie diese Methode beim Start auf, bevor eine weitere Initialisierung im Zusammenhang mit erfolgt. Amazon GameLift Servers Für diese Methode sind explizite Serverparameter erforderlich, um die Kommunikation zwischen dem Server und dem Amazon GameLift Servers Dienst einzurichten. Sie verwendet ein Idempotenz-Token, sodass Sie diesen Aufruf sicher wiederholen können, wenn er fehlschlägt.

Syntax

```
GenericOutcome InitSDK(ServerParameters serverParameters);
```

Parameters

[ServerParameters](#)

Um einen Spieleserver auf einer Amazon GameLift Servers Anywhere-Flotte zu initialisieren, konstruieren Sie ein `ServerParameters` Objekt mit den folgenden Informationen:

- Die URL, mit der die WebSocket Verbindung zu Ihrem Spieleserver hergestellt wurde.
- Die ID des Prozesses, der zum Hosten deines Spieleservers verwendet wird.
- Die ID des Computers, auf dem deine Gameserver-Prozesse gehostet werden.
- Die ID der Amazon GameLift Servers Flotte, die Ihren Amazon GameLift Servers Anywhere-Computer enthält.
- Das durch den Amazon GameLift Servers Vorgang generierte Autorisierungstoken.

Rückgabewert

Bei Erfolg wird ein `InitSdkOutcome` Objekt zurückgegeben, das anzeigt, dass der Serverprozess aufrufbereit ist [ProcessReady\(\)](#).

 Note

Wenn Aufrufe von für Spiele-Builds, die auf Anywhere-Flotten bereitgestellt werden, fehlschlagen, überprüfen Sie den `ServerSdkVersion` Parameter, der beim Erstellen der Build-Ressource verwendet wurde. `InitSDK()` Sie müssen diesen Wert explizit auf die verwendete Server-SDK-Version festlegen. Der Standardwert für diesen Parameter ist 4.x, was nicht kompatibel ist. Um dieses Problem zu beheben, erstellen Sie einen neuen Build und stellen Sie ihn in einer neuen Flotte bereit.

Beispiel

```
//Define the server parameters
string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
string processId = "PID1234";
string fleetId = "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
string hostId = "HardwareAnywhere";
string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
ServerParameters serverParameters =
    new ServerParameters(webSocketUrl, processId, hostId, fleetId, authToken);

//Call InitSDK to establish a local connection with the GameLift agent to enable
further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
```

ProcessReady()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess bereit ist, Spielsitzungen abzuhalten. Rufen Sie diese Methode nach dem Aufrufen auf [InitSDK\(\)](#). Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

Parameters

[ProcessParameters](#)

Ein `ProcessParameters` Objekt enthält Informationen über den Serverprozess.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel veranschaulicht sowohl die Methoden- als auch die Delegatenfunktionsimplementierung.

```
// Set parameters and call ProcessReady
ProcessParameters processParams = new ProcessParameters(
    this.OnStartGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnUpdateGameSession,
    port,
    new LogParameters(new List<string>()
// Examples of log and error files written by the game server
{
    "C:\\game\\logs",
    "C:\\game\\error"
}))
);
GenericOutcome processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

ProcessEnding()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess beendet wird. Ruft diese Methode nach allen anderen Bereinigungsaufgaben (einschließlich des Herunterfahrens der aktiven Spielsitzung) und vor dem Beenden des Vorgangs auf. Je nach Ergebnis von wird der ProcessEnding() Prozess mit Erfolg (0) oder Fehler (-1) beendet und generiert ein Flottenereignis. Wenn der Prozess mit einem Fehler beendet wird, ist das generierte Flottenereignis SERVER_PROCESS_TERMINATED_UNHEALTHY

Syntax

```
GenericOutcome ProcessEnding()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird `ProcessEnding()` der Serverprozess aufgerufen, `Destroy()` bevor er mit einem Exit-Code für Erfolg oder Fehler beendet wird.

```
GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();
GameLiftServerAPI.Destroy();

if (processEndingOutcome.Success)
{
    Environment.Exit(0);
}
else
{
    Console.WriteLine("ProcessEnding() failed. Error: " +
        processEndingOutcome.Error.ToString());
    Environment.Exit(-1);
}
```

ActivateGameSession()

Meldet, Amazon GameLift Servers dass der Serverprozess eine Spielsitzung aktiviert hat und nun bereit ist, Spielerverbindungen zu empfangen. Diese Aktion sollte als Teil der `onStartGameSession()` Callback-Funktion aufgerufen werden, nachdem alle Spielsitzungen initialisiert wurden.

Syntax

```
GenericOutcome ActivateGameSession()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird `ActivateGameSession()` als Teil der `onStartGameSession()`-Delegate-Funktion aufgerufen.

```
void OnStartGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    // When ready to receive players
    GenericOutcome activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spiellersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt.

Syntax

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy
playerSessionPolicy)
```

Parameters

playerSessionPolicy

Zeichenkettenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Gültige Werte sind:

- `ACCEPT_ALL` – Akzeptiert alle neuen Spiellersitzungen.
- `DENY_ALL` – Verwehrt neue Spiellersitzungen.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
GenericOutcome updatePlayerSessionPolicyOutcome =  
    GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die vom aktiven Serverprozess gehostet wird.

Bei Prozessen im Leerlauf, die während einer Spielsitzung nicht aktiviert wurden, gibt der Aufruf a [the section called “GameLiftError”](#) zurück.

Syntax

```
AwsStringOutcome GetGameSessionId()
```

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als [the section called “AwsStringOutcome”](#)-Objekt zurück. Wenn das nicht erfolgreich ist, wird eine Fehlermeldung zurückgegeben.“

Beispiel

```
AwsStringOutcome getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess führt diese Aktion aus, nachdem er einen `onProcessTerminate()` Rückruf von Amazon GameLift Servers erhalten hat. Amazon GameLift Servers ruft `onProcessTerminate()` aus den folgenden Gründen auf:

- Wenn der Serverprozess einen schlechten Zustand gemeldet hat oder nicht darauf reagiert hatAmazon GameLift Servers.
- Wenn die Instanz während eines Scale-Down-Ereignisses beendet wird.

- [Wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.](#)

Syntax

```
AwsDateTimeOutcome GetTerminationTime()
```

Rückgabewert

Bei Erfolg wird die Kündigungszeit als Objekt [the section called “AwsDateTimeOutcome”](#) zurückgegeben. Der Wert ist die Endzeit, ausgedrückt in verstrichenen Ticks, die seitdem vergangen sind. 0001 00:00:00 Beispielsweise entspricht der Wert für Datum und Uhrzeit den 2020-09-13 12:26:40 -0000Z Ticks. 6373559680000000000 Wenn keine Kündigungszeit verfügbar ist, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
AwsDateTimeOutcome getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

AcceptPlayerSession()

Benachrichtigt Amazon GameLift Servers, dass ein Spieler mit der angegebenen Spielsitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Bestätigung benötigt. Amazon GameLift Servers überprüft, ob die Player-Sitzungs-ID gültig ist. Nachdem die Spielsitzung bestätigt wurde, wird der Status des Spieler-Slots von RESERVIERT auf AKTIV Amazon GameLift Servers geändert.

Syntax

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Parameters

playerSessionId

Eindeutige ID GameLift, die ausgestellt wird, wenn eine neue Spielsitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel zeigt eine Funktion zur Bearbeitung einer Verbindungsanfrage, einschließlich der Validierung und Ablehnung einer ungültigen Spielsitzung. IDs

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerSessionId)
{
    GenericOutcome acceptPlayerSessionOutcome =
    GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
    if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);
    }
}
```

RemovePlayerSession()

Benachrichtigt Amazon GameLift Servers, dass ein Spieler die Verbindung zum Serverprozess getrennt hat. Amazon GameLift Servers ändert daraufhin den Player-Slot auf „Verfügbar“.

Syntax

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Parameters

playerSessionId

Eindeutige ID Amazon GameLift Servers, die ausgestellt wird, wenn eine neue Spielsitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
GenericOutcome removePlayerSessionOutcome =  
    GameLiftServerAPI.RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

Ruft Spielersitzungsdaten ab, einschließlich Einstellungen, Sitzungsmetadaten und Spielerdaten. Verwenden Sie diese Aktion, um für eine einzelne Spielersitzung, für alle Spielersitzungen in einer Spielsitzung oder für alle Spielersitzungen zu einer einzelnen Spieler-ID abzurufen.

Syntax

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest  
    describePlayerSessionsRequest)
```

Parameters

[DescribePlayerSessionsRequest](#)

Ein [the section called “DescribePlayerSessionsRequest”](#) Objekt, das beschreibt, welche Spielersitzungen abgerufen werden sollen.

Rückgabewert

Bei Erfolg wird ein [the section called “DescribePlayerSessionsOutcome”](#) Objekt zurückgegeben, das eine Reihe von Spielersitzungsobjekten enthält, die den Anforderungsparametern entsprechen.

Beispiel

Dieses Beispiel zeigt eine Anforderung für alle Spielersitzungen, die derzeit aktiv mit einer angegebenen Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen NextToken und ihn auf 10 setzen, Amazon GameLift Servers werden die Sitzungsdatensätze der ersten 10 Spieler zurückgegeben, die der Anfrage entsprechen.

```
// Set request parameters  
DescribePlayerSessionsRequest describePlayerSessionsRequest = new  
    DescribePlayerSessionsRequest()  
{
```

```
GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    //gets the ID for the
current game session
Limit = 10,
PlayerSessionStatusFilter =
    PlayerSessionStatusMapper.GetNameForPlayerSessionStatus(PlayerSessionStatus.ACTIVE)
};
// Call DescribePlayerSessions
DescribePlayerSessionsOutcome describePlayerSessionsOutcome =
    GameLiftServerAPI.DescribePlayerSessions(describePlayerSessionsRequest);
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Weitere Informationen finden Sie unter [FlexMatchBackfill-Funktion](#).

Diese Aktion ist asynchron. Wenn neue Spieler gematcht werden, Amazon GameLift Servers liefert es mithilfe der Callback-Funktion aktualisierte Matchmaker-Daten. `OnUpdateGameSession()`

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst [StopMatchBackfill\(\)](#) auf, um die ursprüngliche Anforderung abzuberechnen.

Syntax

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
startBackfillRequest);
```

Parameters

[StartMatchBackfillRequest](#)

Ein `StartMatchBackfillRequest` Objekt enthält Informationen über die Backfill-Anfrage.

Rückgabewert

Gibt ein [the section called "StartMatchBackfillOutcome"](#) Objekt mit der passenden Backfill-Ticket-ID oder einem Fehler mit einer Fehlermeldung zurück.

Beispiel

```
// Build a backfill request
```

```

StartMatchBackfillRequest startBackfillRequest = new StartMatchBackfillRequest()
{
    TicketId = "1111aaaa-22bb-33cc-44dd-5555eeee66ff", //optional
    MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    // gets ID for
current game session
    MatchmakerData matchmakerData =
        MatchmakerData.FromJson(gameSession.MatchmakerData), // gets matchmaker data for
current players
    // get matchmakerData.Players
    // remove data for players who are no longer connected
    Players = ListOfPlayersRemainingInTheGame
};

// Send backfill request
StartMatchBackfillOutcome startBackfillOutcome =
    GameLiftServerAPI.StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void OnUpdateGameSession(GameSession myGameSession)
{
    // game-specific tasks to prepare for the newly matched players and update matchmaker
data as needed
}

```

StopMatchBackfill()

Bricht eine aktive Match-Backfill-Anfrage ab. Weitere Informationen finden Sie unter [FlexMatchBackfill-Funktion](#).

Syntax

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameters

[StopMatchBackfillRequest](#)

Ein StopMatchBackfillRequest Objekt, das Details zu dem Matchmaking-Ticket enthält, das Sie beenden.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
// Set backfill stop request parameters
StopMatchBackfillRequest stopBackfillRequest = new StopMatchBackfillRequest(){
    TicketId = "1111aaaa-22bb-33cc-44dd-5555eeee66ff", //optional, if not provided one is
    autogenerated
    MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result //gets the ID for the
    current game session
};
GenericOutcome stopBackfillOutcome =
    GameLiftServerAPI.StopMatchBackfillRequest(stopBackfillRequest);
```

GetComputeCertificate()

Ruft den Pfad zum TLS-Zertifikat ab, das zur Verschlüsselung der Netzwerkverbindung zwischen dem Spieleserver und deinem Spielclient verwendet wird. Du kannst den Zertifikatspfad verwenden, wenn du dein Computergerät bei einer Amazon GameLift Servers Anywhere-Flotte registrierst. Weitere Informationen finden Sie unter [RegisterCompute](#).

Syntax

```
GetComputeCertificateOutcome GetComputeCertificate();
```

Rückgabewert

Gibt ein `GetComputeCertificateResponse` Objekt zurück, das Folgendes enthält:

- `CertificatePath`: Der Pfad zum TLS-Zertifikat auf Ihrer Rechenressource. Wenn Sie eine Amazon GameLift Servers verwaltete Flotte verwenden, enthält dieser Pfad:
 - `certificate.pem`: Das Endbenutzerzertifikat. Die vollständige Zertifikatskette ist die Kombination aus den an dieses Zertifikat `certificateChain.pem` angehängten Zertifikaten.
 - `certificateChain.pem`: Die Zertifikatskette, die das Stammzertifikat und die Zwischenzertifikate enthält.

- `rootCertificate.pem`: Das Stammzertifikat.
- `privateKey.pem`: Der private Schlüssel für das Endbenutzerzertifikat.
- `ComputeName`: Der Name Ihrer Rechenressource.

Beispiel

```
GetComputeCertificateOutcome getComputeCertificateOutcome =  
    GameLiftServerAPI.GetComputeCertificate();
```

GetFleetRoleCredentials()

Ruft Anmeldeinformationen für IAM-Rollen ab, die Amazon GameLift Servers zur Interaktion mit anderen berechtigen. AWS-Services Weitere Informationen finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).

Syntax

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest  
    request);
```

Parameters

[GetFleetRoleCredentialsRequest](#)

Rollenanmeldedaten, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen auf den Spieleserver erweitern.

Rückgabewert

Gibt ein [the section called “GetFleetRoleCredentialsOutcome”](#)-Objekt zurück.

Beispiel

```
// form the fleet credentials request  
GetFleetRoleCredentialsRequest getFleetRoleCredentialsRequest = new  
    GetFleetRoleCredentialsRequest(){  
    RoleArn = "arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction"  
};
```

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentialsOutcome credentials =  
    GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Zerstören ()

Befreit das Amazon GameLift Servers Spieleserver-SDK aus dem Speicher. Es hat sich bewährt, diese Methode nach `ProcessEnding()` und vor dem Beenden des Prozesses aufzurufen. Wenn Sie eine Anywhere-Flotte verwenden und Serverprozesse nicht nach jeder Spielsitzung beenden, rufen Sie auf `Destroy()` und starten Sie dann `InitSDK()` erneut, bevor Sie benachrichtigen, Amazon GameLift Servers dass der Prozess bereit ist, eine Spielsitzung mit zu hosten. `ProcessReady()`

Syntax

```
GenericOutcome Destroy()
```

Rückgabewert

Gibt ein generisches Ergebnis, bestehend aus Erfolg oder Misserfolg, mit einer Fehlermeldung zurück.

Beispiel

```
// Operations to end game sessions and the server process  
GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();  
  
// Shut down and destroy the instance of the GameLift Game Server SDK  
GenericOutcome destroyOutcome = GameLiftServerAPI.Destroy();  
  
// Exit the process with success or failure  
if (processEndingOutcome.Success)  
{  
    Environment.Exit(0);  
}  
else  
{  
    Console.WriteLine("ProcessEnding() failed. Error: " +  
        processEndingOutcome.Error.ToString());  
    Environment.Exit(-1);  
}
```

Go-Server-SDK für Amazon GameLift Servers -- Aktionen

Verwenden Sie die Server-SDK 5.x-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrieren. Amazon GameLift Servers Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

`GameLiftServerAPI`.godefiniert die SDK-Aktionen des Go-Servers.

Go-Server-SDK für Amazon GameLift Servers -- Datentypen

Verwenden Sie die Server-SDK-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrierenAmazon GameLift Servers. Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

`GameLiftServerAPI`.godefiniert die SDK-Aktionen des Go-Servers.

[Go-Server-SDK für Amazon GameLift Servers -- Aktionen](#)

Datentypen

- [LogParameters](#)
- [MetricsParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)

LogParameters

Ein Objekt, das Dateien identifiziert, die während einer Spielsitzung generiert wurden und die Sie nach dem Ende der Spielsitzung hochladen und speichern möchtenAmazon GameLift Servers. Der Spieleserver stellt `LogParameters` es Amazon GameLift Servers als Teil eines `ProcessParameters` Objekts in einem [ProcessReady\(\)](#) Aufruf bereit.

Eigenschaften	Beschreibung
LogPaths	Die Liste der Verzeichnispfade zu den Protokolldateien des Spieleservers, die Sie für den future Amazon GameLift Servers Zugriff speichern möchten. Der Serverprozess generiert diese Dateien während jeder Spielsitzung. Du definierst Dateipfade und -namen auf deinem Spieleserver und speicherst sie im Hauptverzeichnis des Spiels. Die Protokollpfade müssen absolut sein. Wenn Ihr Spielbuild beispielsweise Spielsitzungsprotokolle in einem Pfad wie <code>speichertMyGame\sessionLogs\</code> , dann wäre der Pfad <code>c:\game\MyGame\sessionLogs</code> auf einer Windows-Instanz. Typ: <code>[]string</code> Required: No

MetricsParameters

Ein Objekt, das Konfigurationsparameter für die Initialisierung des Metriksystems übermittelt. Diese Konfiguration wird verwendet, um StatsD-Berichte, Absturzberichte und das Verarbeitungsverhalten von Metriken einzurichten. Der Spieleserver stellt dies `MetricsParameters` im Amazon GameLift Servers Rahmen eines [InitMetrics\(\)](#) Aufrufs bereit.

Eigenschaften	Beschreibung
StatsdHost	Der StatsD-Serverhost für die Berichterstattung über Metriken (z. B. „localhost“). Typ: <code>string</code> Erforderlich: Ja
StatsdPort	Der StatsD-Serverport für die Metrikberichterstattung (z. B. 8125). Typ: <code>int</code> Erforderlich: Ja

CrashReporterHost	Der Crash-Reporter-Host für die Absturzverfolgung und Prozessüberwachung. Typ: <code>string</code> Erforderlich: Ja
CrashReporterPort	Der Crash-Reporter-Port für die Absturzverfolgung und Prozessüberwachung. Typ: <code>int</code> Erforderlich: Ja
FlushIntervalMs	Das Aktualisierungsintervall der Metriken wird in Millisekunden angegeben. Steuert, wie oft Metriken an StatsD gesendet werden. Typ: <code>int</code> Erforderlich: Ja
MaxPacketSize	Die maximale Paketgröße für Metriken in Byte. Beschränkt die Größe der UDP-Pakete, die an StatsD gesendet werden. Typ: <code>int</code> Erforderlich: Ja

ProcessParameters

Ein Objekt, das die Kommunikation zwischen einem Serverprozess und Amazon GameLift Servers beschreibt. Der Serverprozess stellt diese Informationen Amazon GameLift Servers mit einem Aufruf von bereit [ProcessReady\(\)](#).

Eigenschaften	Beschreibung
LogParameters	Ein Objekt mit Verzeichnispfaden zu Dateien, die während einer Spielsitzung generiert werden. Amazon GameLift Servers kopiert und speichert die Dateien für den future Zugriff. Typ: LogParameters

	Required: No
OnHealthCheck	Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift Servers ruft diese Funktion alle 60 Sekunden auf und wartet 60 Sekunden auf eine Antwort. Der Serverprozess kehrt zurück, TRUE wenn er fehlerfrei ist, FALSE wenn nicht. Wenn keine Antwort zurückgegeben wird, wird der Amazon GameLift Servers Serverprozess als fehlerhaft aufgezeichnet. Typ: OnHealthCheck func() bool Required: No
OnProcessTerminate	Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um das Herunterfahren des Serverprozesses zu erzwingen. Amazon GameLift Servers wartet nach dem Aufrufen dieser Funktion 5 Minuten, bis der Serverprozess heruntergefahren ist, und antwortet mit einem ProcessEnding() Aufruf, bevor er den Serverprozess herunterfährt. Typ: OnProcessTerminate func() Erforderlich: Ja
OnStartGameSession	Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift Servers ruft diese Funktion auf, wenn eine Match-Backfill-Anfrage bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Sie übergibt ein GameSession Objekt, ein Status-Update (updateReason) und die Match-Backfill-Ticket-ID. Typ: OnStartGameSession func (model.GameSession) Erforderlich: Ja

OnUpdateGameSession	Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um aktualisierte Informationen zur Spielsitzung an den Serverprozess weiterzuleiten. Amazon GameLift Servers ruft diese Funktion nach der Verarbeitung einer Match-Backfill-Anfrage auf, um aktualisierte Matchmaker-Daten bereitzustellen. Typ: <code>OnUpdateGameSession func (model.UpdateGameSession)</code> Required: No
Port	Die Portnummer, auf der der Serverprozess nach neuen Spielerverbindungen sucht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden. Typ: <code>int</code> Erforderlich: Ja

UpdateGameSession

Die Aktualisierungen eines Spielsitzungsobjekts, das den Grund für die Aktualisierung der Spielsitzung und die zugehörige Backfill-Ticket-ID enthält, falls Backfill verwendet wird, um Spielersitzungen in der Spielsitzung zu füllen.

Eigenschaften	Beschreibung
GameSession	Ein GameSession-Objekt. Das <code>GameSession</code> Objekt enthält Eigenschaften, die eine Spielsitzung beschreiben. Typ: <code>GameSession GameSession()</code> Erforderlich: Ja
UpdateReason	Der Grund, warum die Spielsitzung aktualisiert wird. Typ: <code>UpdateReason UpdateReason()</code>

Eigenschaften	Beschreibung
	Erforderlich: Ja
BackfillTicketId	Die ID des Backfill-Tickets, das versucht, die Spielsitzung zu aktualisieren. Typ: <code>String</code> Required: No

GameSession

Die Details einer Spielsitzung.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Eine Spielsitzung Amazon Resource Name (ARN) hat das folgende Format: <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Typ: <code>String</code> Required: No
Name	Eine beschreibende Bezeichnung der Spielsitzung. Typ: <code>String</code> Required: No
FleetId	Eine eindeutige Kennung für die Flotte, auf der die Spielsitzung läuft. Typ: <code>String</code> Required: No
MaximumPlayerSessionCount	Die maximale Anzahl von Spielerverbindungen zur Spielsitzung. Typ: <code>Integer</code>

Eigenschaften	Beschreibung
	Required: No
Port	Die Portnummer für die Spielsitzung. Um eine Verbindung zu einem Amazon GameLift Servers Spieleserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: <code>Integer</code> Required: No
IpAddress	Die IP-Adresse der Spielsitzung. Um eine Verbindung zu einem Amazon GameLift Servers Spieleserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: <code>String</code> Required: No
GameSessionData	Eine Reihe von benutzerdefinierten Spielsitzungseigenschaften, die als einzelner Zeichenfolgenwert formatiert sind. Typ: <code>String</code> Required: No
MatchmakerData	Die Informationen über den Matchmaking-Prozess, der zur Erstellung der Spielsitzung verwendet wurde, in JSON-Syntax, formatiert als Zeichenfolge. Zusätzlich zur verwendeten Matchmaking-Konfiguration enthält es Daten zu allen Spielern, die dem Spiel zugewiesen wurden, einschließlich Spielerattributen und Teamzuweisungen. Typ: <code>String</code> Required: No

Eigenschaften	Beschreibung
GameProperties	Eine Reihe von benutzerdefinierten Eigenschaften für eine Spielsitzung, formatiert als Schlüssel/Wert-Paare. Diese Eigenschaften werden zusammen mit der Aufforderung, eine neue Spielsitzung zu starten, übergeben. Typ: <code>map[string] string</code> Required: No
DnsName	Die DNS-ID, die der Instanz zugewiesen ist, auf der die Spielsitzung ausgeführt wird. Werte haben das folgende Format: • TLS-fähige Flotten: <code><unique identifier>.<region identifier>.amazongamelift.com</code> • Non-TLS-enabled Flotten: <code>ec2-<unique identifier>.compute.amazonaws.com</code> Wenn du eine Verbindung zu einer Spielsitzung herstellst, die auf einer TLS-fähigen Flotte läuft, musst du den DNS-Namen verwenden, nicht die IP-Adresse. Typ: <code>String</code> Required: No

ServerParameters

Informationen, die zur Aufrechterhaltung der Verbindung zwischen einem Amazon GameLift Servers Anywhere-Server und dem Amazon GameLift Servers Dienst verwendet werden. Diese Informationen werden verwendet, wenn neue Serverprozesse mit gestartet [InitSDK\(\)](#) werden. Verwenden Sie für Server, die auf Amazon GameLift Servers verwalteten EC2-Instances gehostet werden, ein leeres Objekt.

Eigenschaften	Beschreibung
WebSocket URL	Das <code>GameLiftServerSdkEndpoint</code> Amazon GameLift Servers kehrt zurück, wenn Sie RegisterCompute für eine Amazon GameLift Servers Anywhere-Rechenressource suchen. Typ: <code>string</code> Erforderlich: Ja
ProcessID	Eine eindeutige Kennung, die für den Serverprozess registriert ist, der Ihr Spiel hostet. Typ: <code>string</code> Erforderlich: Ja
HostID	Die eindeutige Kennung der Rechenressource, die den neuen Serverprozess hostet. Das <code>HostID</code> ist die, die bei der Registrierung Ihres Computers <code>ComputeName</code> verwendet wurde. Weitere Informationen finden Sie unter RegisterCompute. Typ: <code>string</code> Erforderlich: Ja
FleetID	Die eindeutige Kennung der Flotte, für die die Berechnung registriert ist. Weitere Informationen finden Sie unter RegisterCompute. Typ: <code>string</code> Erforderlich: Ja
AuthToken	Das dadurch Amazon GameLift Servers generierte Authentifizierungstoken authentifiziert Ihren Amazon GameLift Servers Server. Weitere Informationen finden Sie unter GetComputeAuthToken. Typ: <code>string</code>

Eigenschaften	Beschreibung
	Erforderlich: Ja

StartMatchBackfillRequest

Informationen, die zur Erstellung einer Matchmaking-Backfill-Anfrage verwendet wurden. Der Spieleserver übermittelt diese Informationen Amazon GameLift Servers in einem Anruf an.

[StartMatchBackfill\(\)](#)

Eigenschaften	Beschreibung
GameSessionArn	Die eindeutige Kennung der Spielsitzung. Die API-Operation GetGameSessionId gibt den Bezeichner im ARN-Format zurück. Typ: <code>String</code> Erforderlich: Ja
MatchmakingConfigurationArn	Die eindeutige Kennung (in Form eines ARN), die der Matchmaker für diese Anfrage verwenden soll. Der Matchmaker-ARN für die ursprüngliche Spielsitzung befindet sich im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft . Weitere Informationen zu Matchmaker-Daten finden Sie unter Arbeiten mit Matchmaker-Daten. Typ: <code>String</code> Erforderlich: Ja
Players	Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Typ: <code>[]model.Player</code> Erforderlich: Ja
TicketId	Die eindeutige Kennung für ein Matchmaking- oder Match-Backfill-Anfrage-ticket. Amazon GameLift ServersGeneriert einen, wenn Sie keinen Wert

Eigenschaften	Beschreibung
	angeben. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abubrechen. Typ: <code>String</code> Required: No

Player

Das Objekt, das einen Spieler beim Matchmaking darstellt. Wenn eine Matchmaking-Anfrage gestartet wird, hat ein Spieler eine Spieler-ID, Attribute und möglicherweise Latenzdaten. Amazon GameLift Servers fügt Teaminformationen hinzu, nachdem ein Spiel gespielt wurde.

Eigenschaften	Beschreibung
LatencyInMS	Eine Reihe von Werten, ausgedrückt in Millisekunden, die die Latenz angeben, die ein Spieler erlebt, wenn er mit einem Standort verbunden ist. Wenn diese Eigenschaft verwendet wird, wird der Player nur für die aufgelisteten Standorte abgeglichen. Wenn ein Matchmaker eine Regel hat, die die Latenz der Spieler auswertet, müssen die Spieler die zu vergleichende Latenz melden. Typ: <code>map[string] int</code> Required: No
PlayerAttributes	Eine Sammlung von Schlüssel/Wert-Paaren, die Spielerinformationen zur Verwendung bei der Spielsuche enthalten. Die Schlüssel der Spielerattribute müssen mit den in einem Matchmaking-Regelsatz PlayerAttributes verwendeten übereinstimmen. Weitere Informationen zu Spielerattributen finden Sie unter AttributeValue. Typ: <code>map[string] AttributeValue</code> Required: No

Eigenschaften	Beschreibung
PlayerId	Eine eindeutige Kennung für einen Spieler. Typ: <code>String</code> Required: No
Team	Der Name der Mannschaft, der der Spieler in einem Spiel zugewiesen wird. Sie definieren den Teamnamen im Matchmaking-Regelsatz. Typ: <code>String</code> Required: No

DescribePlayerSessionsRequest

Ein Objekt, das angibt, welche Spilersitzungen abgerufen werden sollen. Der Serverprozess stellt diese Informationen mit einem [DescribePlayerSessions\(\)](#) Aufruf von bereitAmazon GameLift Servers.

Eigenschaften	Beschreibung
GameSessionID	Eine eindeutige Kennung für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spilersitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code>. Das <code>GameSessionID</code> ist eine benutzerdefinierte ID-Zeichenfolge oder eine generierte Zeichenfolge. Typ: <code>String</code> Required: No
PlayerSessionID	Die eindeutige Kennung für eine Spilersitzung. Verwenden Sie diesen Parameter, um eine einzelne spezifische Spilersitzung anzufordern. Typ: <code>String</code>

Eigenschaften	Beschreibung
	Required: No
PlayerID	Die eindeutige Kennung für einen Spieler. Verwenden Sie diesen Parameter , um alle Spielsitzungen für einen bestimmten Spieler anzufordern. Siehe Spieler generieren IDs. Typ: String Required: No
PlayerSessionStatusFilter	Der Sitzungsstatus des Spielers, nach dem die Ergebnisse gefiltert werden sollen. Zu den möglichen Sitzungsstatus eines Spielers gehören: • RESERVIERT — Die Anfrage für die Spielsitzung wurde empfangen, aber der Spieler hat keine Verbindung zum Serverprozess hergestellt oder wurde nicht validiert. • AKTIV — Der Spieler wurde durch den Serverprozess validiert und ist verbunden. • ABGESCHLOSSEN — Die Spielerverbindung wurde unterbrochen. • TIMEDOUT — Eine Anfrage für eine Spielsitzung wurde empfangen, aber der Spieler hat keine Verbindung hergestellt oder wurde nicht innerhalb des Timeoutlimits (60 Sekunden) bestätigt. Typ: String Required: No
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite anzeigt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Spielsitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: String Required: No

Eigenschaften	Beschreibung
Limit	Die maximale Anzahl der auszugebenden Ergebnisse. Wenn Sie eine Spielsitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: <code>int</code> Required: No

StopMatchBackfillRequest

Informationen, die verwendet werden, um eine Matchmaking-Backfill-Anfrage zu stornieren. Der Spieleserver übermittelt diese Informationen in einem Anruf an den Amazon GameLift Servers Service. [StopMatchBackfill\(\)](#)

Eigenschaften	Beschreibung
GameSessionArn	Die eindeutige Spielsitzungs-ID der Anfrage, die storniert wurde. Typ: <code>string</code> Required: No
MatchmakingConfigurationArn	Die eindeutige Kennung des Matchmakers, an den diese Anfrage gesendet wurde. Typ: <code>string</code> Required: No
TicketId	Die eindeutige Kennung des Tickets für die Backfill-Anfrage, das storniert werden soll. Typ: <code>string</code> Required: No

GetFleetRoleCredentialsRequest

Die Zugangsdaten für die Rolle, mit denen der eingeschränkte Zugriff auf deine AWS Ressourcen auf den Spielserver ausgedehnt wird. Weitere Informationen finden Sie unter [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#).

Eigenschaften	Beschreibung
RoleArn	Der ARN der Servicerolle, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen erweitert. Typ: <code>string</code> Erforderlich: Ja
RoleSessionName	Der Name der Sitzung, die die Verwendung der Rollenmeldedaten beschreibt. Typ: <code>string</code> Erforderlich: Ja

[Go-Server-SDK für Amazon GameLift Servers -- Datentypen](#)

Aktionen

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [InitMetricsFromEnvironment\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)

- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Zerstören \(\)](#)

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
func GetSdkVersion() (string, error)
```

Rückgabewert

Gibt bei Erfolg die aktuelle SDK-Version als Zeichenfolge zurück. Die zurückgegebene Zeichenfolge enthält die Versionsnummer (Beispiel `5.0.0`). Wenn dies nicht erfolgreich ist, wird eine Fehlermeldung wie `zurückgegebencommon.SdkVersionDetectionFailed` zurückgegeben.

Beispiel

```
version, err := server.GetSdkVersion()
```

InitMetrics()

Initialisiert die Erfassung von Metriken für das Amazon GameLift Servers SDK. Mit dieser Methode werden Berichte über Metriken eingerichtet, um die Serverleistung und den Zustand zu überwachen. Rufen Sie diese Methode danach [InitSDK\(\)](#), aber vorher auf [ProcessReady\(\)](#).

Syntax

```
func InitMetrics() error  
func InitMetrics(metricsParameters MetricsParameters) error
```

Parameters

MetricsParameters (Optional)

Ein `MetricsParameters` Objekt, das die Erfassung von Metriken konfiguriert. Falls nicht angegeben, wird die Standardkonfiguration für Metriken verwendet. Die `MetricsParameters` Struktur enthält die folgenden Felder:

- `StatsdHost`- Der Hostname oder die IP-Adresse des StatsD-Servers.
- `StatsdPort`- Die Portnummer für den StatsD-Server.
- `CrashReporterHost`- Der Hostname oder die IP-Adresse des Crash-Reporter-Dienstes.
- `CrashReporterPort`- Die Portnummer für den Crash-Reporter-Dienst.
- `FlushIntervalMs`- Das Intervall in Millisekunden für das Löschen von Metrikdaten.
- `MaxPacketSize`- Die maximale Größe von Metrikpaketen in Byte.

Weitere Informationen zur `MetricsParameters` Struktur finden Sie unter [Server-SDK 5.x für C#-Datentypen](#).

Rückgabewert

Bei Erfolg wird ein `nil` Fehler zurückgegeben, der darauf hinweist, dass die Metrikerfassung erfolgreich initialisiert wurde.

Beispiel

Initialisieren Sie Metriken mit der Standardkonfiguration:

```
err := server.InitMetrics()
```

Metriken mit benutzerdefinierter Konfiguration initialisieren:

```
metricsParams := MetricsParameters{
    StatsdHost:      "localhost",
    StatsdPort:      8125,
    CrashReporterHost: "localhost",
    CrashReporterPort: 9125,
    FlushIntervalMs:  5000,
    MaxPacketSize:    1024,
}
```

```
err := server.InitMetrics(metricsParams)
```

InitMetricsFromEnvironment()

Initialisiert die Erfassung von Metriken für das Amazon GameLift Servers SDK mithilfe der Konfiguration aus Umgebungsvariablen. Diese Methode richtet die Metrikberichterstattung mithilfe von Standardeinstellungen ein, die von der Laufzeitumgebung abgeleitet wurden.

Rufen Sie diese Methode danach [InitSDK\(\)](#), aber vorher auf [ProcessReady\(\)](#).

Syntax

```
func InitMetricsFromEnvironment() error
```

Rückgabewert

Bei Erfolg wird ein `nil` Fehler zurückgegeben, der darauf hinweist, dass die Metrikerfassung mithilfe der Umgebungskonfiguration erfolgreich initialisiert wurde.

Beispiel

```
err := server.InitMetricsFromEnvironment()
```

InitSDK()

Initialisiert das Amazon GameLift Servers-SDK. Rufen Sie diese Methode beim Start auf, bevor eine weitere Initialisierung im Zusammenhang Amazon GameLift Servers mit erfolgt. Diese Methode richtet die Kommunikation zwischen dem Server und dem Amazon GameLift Servers Dienst ein.

Syntax

```
func InitSDK(params ServerParameters) error
```

Parameters

[ServerParameters](#)

Um einen Spieleserver auf einer Amazon GameLift Servers Anywhere-Flotte zu initialisieren, konstruieren Sie ein `ServerParameters` Objekt mit den folgenden Informationen:

- Die URL, mit der die WebSocket Verbindung zu Ihrem Spieleserver hergestellt wurde.

- Die ID des Prozesses, der zum Hosten deines Spieleservers verwendet wird.
- Die ID des Computers, auf dem deine Gameserver-Prozesse gehostet werden.
- Die ID der Amazon GameLift Servers Flotte, die Ihren Amazon GameLift Servers Anywhere-Computer enthält.
- Das durch den Amazon GameLift Servers Vorgang generierte Autorisierungstoken.

Um einen Spieleserver auf einer Amazon GameLift Servers verwalteten EC2-Flotte zu initialisieren, konstruieren Sie ein `ServerParameters` Objekt ohne Parameter. Mit diesem Aufruf richtet der Amazon GameLift Servers Agent die Rechenumgebung ein und stellt automatisch eine Verbindung zum Amazon GameLift Servers Dienst für Sie her.

Rückgabewert

Bei Erfolg wird ein `nil` Fehler zurückgegeben, der darauf hinweist, dass der Serverprozess aufrufbereit ist [ProcessReady\(\)](#).

Note

Wenn Aufrufe von für Spiel-Builds, die auf Anywhere-Flotten bereitgestellt werden, fehlschlagen, überprüfen Sie den `ServerSdkVersion` Parameter, der beim Erstellen der Build-Ressource verwendet wurde. `InitSDK()` Sie müssen diesen Wert explizit auf die verwendete Server-SDK-Version festlegen. Der Standardwert für diesen Parameter ist 4.x, was nicht kompatibel ist. Um dieses Problem zu beheben, erstellen Sie einen neuen Build und stellen Sie ihn in einer neuen Flotte bereit.

Beispiel

Amazon GameLift ServersIrgendwo, Beispiel

```
//Define the server parameters
serverParameters := ServerParameters {
  WebSocketURL: "wss://us-west-1.api.amazongamelift.com",
  ProcessID: "PID1234",
  HostID: "HardwareAnywhere",
  FleetID: "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa",
  AuthToken: "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
}
```

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
err := server.InitSDK(serverParameters)
```

Amazon GameLift Servers Beispiel für verwaltetes EC2

```
//Define the server parameters
serverParameters := ServerParameters {}

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
err := server.InitSDK(serverParameters)
```

ProcessReady()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess bereit ist, Spielsitzungen zu hosten. Rufen Sie diese Methode nach dem Aufrufen auf [InitSDK\(\)](#). Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
func ProcessReady(param ProcessParameters) error
```

Parameters

ProcessParameters

Ein [ProcessParameters](#) Objekt übermittelt die folgenden Informationen über den Serverprozess:

- Die Namen der im Spielservercode implementierten Callback-Methoden, die der Amazon GameLift Servers Dienst aufruft, um mit dem Serverprozess zu kommunizieren.
- Die Portnummer, die der Serverprozess abhört.
- Der [LogParameters](#) Datentyp, der den Pfad zu allen spielsitzungsspezifischen Dateien enthält, die Sie erfassen und Amazon GameLift Servers speichern möchten.

Rückgabewert

Gibt einen Fehler mit einer Fehlermeldung zurück, wenn die Methode fehlschlägt. Gibt zurück `nil`, ob die Methode erfolgreich ist.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des [ProcessReady\(\)](#)-Aufrufs und der Delegate-Funktion.

```
// Define the process parameters
processParams := ProcessParameters {
    OnStartGameSession: gameProcess.OnStartGameSession,
    OnUpdateGameSession: gameProcess.OnGameSessionUpdate,
    OnProcessTerminate: gameProcess.OnProcessTerminate,
    OnHealthCheck: gameProcess.OnHealthCheck,
    Port: port,
    LogParameters: LogParameters {    // logging and error example
        []string {"C:\\game\\logs", "C:\\game\\error"}
    }
}

err := server.ProcessReady(processParams)
```

ProcessEnding()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess beendet wird. Ruft diese Methode nach allen anderen Bereinigungsaufgaben (einschließlich des Herunterfahrens der aktiven Spielsitzung) und vor dem Beenden des Vorgangs auf. Je nach Ergebnis von wird der ProcessEnding() Prozess mit Erfolg (0) oder Fehler (-1) beendet und generiert ein Flottenereignis. Wenn der Prozess mit einem Fehler beendet wird, ist das generierte Flottenereignis. SERVER_PROCESS_TERMINATED_UNHEALTHY

Syntax

```
func ProcessEnding() error
```

Rückgabewert

Gibt den Fehlercode 0 oder einen definierten Fehlercode zurück.

Beispiel

```
// operations to end game sessions and the server process
defer func() {
    err := server.ProcessEnding()
```

```
server.Destroy()
if err != nil {
    fmt.Println("ProcessEnding() failed. Error: ", err)
    os.Exit(-1)
} else {
    os.Exit(0)
}
}
```

ActivateGameSession()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess eine Spielsitzung aktiviert hat und nun bereit ist, Spielerverbindungen zu empfangen. Diese Aktion wird als Teil der `onStartGameSession()` Callback-Funktion aufgerufen, nachdem alle Spielsitzungen initialisiert wurden.

Syntax

```
func ActivateGameSession() error
```

Rückgabewert

Gibt einen Fehler mit einer Fehlermeldung zurück, wenn die Methode fehlschlägt.

Beispiel

Dieses Beispiel wird als Teil der `onStartGameSession()` Delegate-Funktion `ActivateGameSession()` aufgerufen.

```
func OnStartGameSession(GameSession gameSession) {
    // game-specific tasks when starting a new game session, such as loading map
    // Activate when ready to receive players
    err := server.ActivateGameSession();
}
```

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt.

Syntax

```
func UpdatePlayerSessionCreationPolicy(policy model.PlayerSessionCreationPolicy) error
```

Parameters

playerSessionCreationRichtlinie

Zeichenkettenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Gültige Werte sind:

- **model.AcceptAll**— Akzeptiert alle Sitzungen mit neuen Spielern.
- **model.DenyAll**— Alle Sitzungen neuer Spieler ablehnen.

Rückgabewert

Gibt einen Fehler mit einer Fehlermeldung zurück, falls ein Fehler auftritt.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
err := server.UpdatePlayerSessionCreationPolicy(model.AcceptAll)
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die vom aktiven Serverprozess gehostet wird.

Syntax

```
func GetGameSessionID() (string, error)
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird die Spielsitzungs-ID zurückgegeben und es wird kein Fehler angezeigt. Bei Prozessen im Leerlauf, die während einer Spielsitzung noch nicht aktiviert wurden, gibt der Aufruf eine leere Zeichenfolge und einen `nil` Fehler zurück.

Beispiel

```
gameSessionID, err := server.GetGameSessionID()
```

GetTerminationTime()

Gibt die Uhrzeit zurück, zu der ein Serverprozess planmäßig heruntergefahren werden soll, sofern eine Kündigungszeit verfügbar ist. Ein Serverprozess führt diese Aktion aus, nachdem er einen `onProcessTerminate()` Rückruf von Amazon GameLift Servers erhalten hat. Amazon GameLift Servers ruft `onProcessTerminate()` aus den folgenden Gründen auf:

- Wenn der Serverprozess einen schlechten Zustand gemeldet hat oder nicht darauf reagiert hatAmazon GameLift Servers.
- Wenn die Instanz während eines Scale-Down-Ereignisses beendet wird.
- [Wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.](#)

Syntax

```
func GetTerminationTime() (int64, error)
```

Rückgabewert

Bei Erfolg wird der Zeitstempel in Epochensekunden zurückgegeben, für den das Herunterfahren des Serverprozesses geplant ist, und es wird ein Fehler beim Beenden angezeigt. `nil` Der Wert ist die Endzeit, ausgedrückt in verstrichenen Ticks von. `0001 00:00:00` Der Wert für Datum und Uhrzeit `2020-09-13 12:26:40 -000Z` entspricht beispielsweise Ticks. `637355968000000000` Wenn keine Kündigungszeit verfügbar ist, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
terminationTime, err := server.GetTerminationTime()
```

AcceptPlayerSession()

Benachrichtigt Amazon GameLift Servers, dass ein Spieler mit der angegebenen Spilersitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Bestätigung benötigt. Amazon GameLift Servers überprüft, ob die Spilersitzungs-ID gültig ist. Nachdem die Spilersitzung bestätigt wurde, Amazon GameLift Servers ändert sich der Status des Spieler-Slots von RESERVED zu ACTIVE.

Syntax

```
func AcceptPlayerSession(playerSessionID string) error
```

Parameters

playerSessionId

Eindeutige ID Amazon GameLift Servers, die ausgestellt wird, wenn eine neue Spilersitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird eine Verbindungsanforderung behandelt, die das Überprüfen und Ablehnen einer ungültigen Spilersitzung beinhaltet. IDs

```
func ReceiveConnectingPlayerSessionID(conn Connection, playerSessionID string) {  
    err := server.AcceptPlayerSession(playerSessionID)  
    if err != nil {  
        connection.Accept()  
    } else {  
        connection.Reject(err.Error())  
    }  
}
```

RemovePlayerSession()

Benachrichtigt Amazon GameLift Servers, dass ein Spieler die Verbindung zum Serverprozess getrennt hat. Amazon GameLift Servers ändert daraufhin den Player-Slot auf verfügbar.

Syntax

```
func RemovePlayerSession(playerSessionID string) error
```

Parameters

playerSessionId

Eindeutige ID Amazon GameLift Servers, die ausgestellt wird, wenn eine neue Spielsitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
err := server.RemovePlayerSession(playerSessionID)
```

DescribePlayerSessions()

Ruft Spielsitzungsdaten ab, einschließlich Einstellungen, Sitzungsmetadaten und Spielerdaten. Verwenden Sie diese Methode, um Informationen zu folgenden Themen abzurufen:

- Eine Einzelspieler-Sitzung
- Alle Spielsitzungen in einer Spielsitzung
- Alle Spielsitzungen, die einer einzelnen Spieler-ID zugeordnet sind

Syntax

```
func DescribePlayerSessions(req request.DescribePlayerSessionsRequest)  
    (result.DescribePlayerSessionsResult, error) {  
    return srv.describePlayerSessions(&req)  
}
```

Parameters

DescribePlayerSessionsRequest

Ein `DescribePlayerSessionsRequest` Objekt beschreibt, welche Spilersitzungen abgerufen werden sollen.

Rückgabewert

Bei Erfolg wird ein `DescribePlayerSessionsResult` Objekt zurückgegeben, das eine Reihe von Spilersitzungsobjekten enthält, die den Anforderungsparametern entsprechen.

Beispiel

In diesem Beispiel werden alle Spilersitzungen angefordert, die aktiv mit einer bestimmten Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen `NextToken` und ihn auf 10 setzen, werden die Sitzungsdatensätze der ersten 10 Spieler Amazon GameLift Servers zurückgegeben, die der Anfrage entsprechen.

```
// create request
describePlayerSessionsRequest := request.NewDescribePlayerSessions()
describePlayerSessionsRequest.GameSessionID, _ = server.GetGameSessionID() // get ID
for the current game session
describePlayerSessionsRequest.Limit = 10 // return the
first 10 player sessions
describePlayerSessionsRequest.PlayerSessionStatusFilter = "ACTIVE" // Get all
player sessions actively connected to the game session

describePlayerSessionsResult, err :=
server.DescribePlayerSessions(describePlayerSessionsRequest)
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Weitere Informationen finden Sie unter [FlexMatchBackfill-Funktion](#).

Diese Aktion ist asynchron. Wenn neue Spieler gematcht werden, Amazon GameLift Servers liefert es mithilfe der Callback-Funktion aktualisierte Matchmaker-Daten. `OnUpdateGameSession()`

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst [StopMatchBackfill\(\)](#) auf, um die ursprüngliche Anforderung abubrechen.

Syntax

```
func StartMatchBackfill(req request.StartMatchBackfillRequest)
    (result.StartMatchBackfillResult, error)
```

Parameters

[StartMatchBackfillRequest](#)

Ein StartMatchBackfillRequest Objekt übermittelt die folgenden Informationen:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Information ist optional. Wenn keine ID angegeben wird, wird eine Amazon GameLift Servers generiert.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert ist in den Matchmaker-Daten der Spielsitzung enthalten.
- Die ID der Spielsitzung, die aufgefüllt werden soll.
- Die verfügbaren Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Rückgabewert

Gibt ein StartMatchBackfillResult Objekt mit der ID des Match-Backfill-Tickets zurück oder schlägt fehl und es wird eine Fehlermeldung angezeigt.

Beispiel

```
// form the request
startBackfillRequest := request.NewStartMatchBackfill()
startBackfillRequest.RequestID = "1111aaaa-22bb-33cc-44dd-5555eeee66ff" // optional
startBackfillRequest.MatchmakingConfigurationArn = "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"
var matchMaker model.MatchmakerData
if err := matchMaker.UnmarshalJSON([]byte(gameSession.MatchmakerData)); err != nil {
    return
}
```

```
startBackfillRequest.Players = matchMaker.Players
res, err := server.StartMatchBackfill(startBackfillRequest)

// Implement callback function for backfill
func OnUpdateGameSession(myGameSession model.GameSession) {
    // game-specific tasks to prepare for the newly matched players and update
    matchmaker data as needed
}
```

StopMatchBackfill()

Bricht eine aktive Match-Backfill-Anfrage ab. Weitere Informationen finden Sie unter [FlexMatchBackfill-Funktion](#).

Syntax

```
func StopMatchBackfill(req request.StopMatchBackfillRequest) error
```

Parameters

[StopMatchBackfillRequest](#)

Ein StopMatchBackfillRequest Objekt, das das zu stornierende Matchmaking-Ticket identifiziert:

- Die der Backfill-Anfrage zugewiesene Ticket-ID.
- Der Matchmaker, an den die Backfill-Anfrage gesendet wurde.
- Die Spielsitzung, die mit der Backfill-Anfrage verknüpft ist.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
stopBackfillRequest := request.NewStopMatchBackfill() // Use this function to create
request
stopBackfillRequest.TicketID = "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
stopBackfillRequest.MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"
```

```
//error  
err := server.StopMatchBackfill(stopBackfillRequest)
```

GetComputeCertificate()

Ruft den Pfad zum TLS-Zertifikat ab, das zur Verschlüsselung der Netzwerkverbindung zwischen dem Spieleserver und deinem Spielclient verwendet wird. Du kannst den Zertifikatspfad verwenden, wenn du dein Computergerät bei einer Amazon GameLift Servers Anywhere-Flotte registrierst. Weitere Informationen finden Sie unter [RegisterCompute](#).

Syntax

```
func GetComputeCertificate() (result.GetComputeCertificateResult, error)
```

Rückgabewert

Gibt ein `GetComputeCertificateResult` Objekt zurück, das Folgendes enthält:

- `CertificatePath`: Der Pfad zum TLS-Zertifikat auf Ihrer Rechenressource. Wenn Sie eine Amazon GameLift Servers verwaltete Flotte verwenden, enthält dieser Pfad:
 - `certificate.pem`: Das Endbenutzerzertifikat. Die vollständige Zertifikatskette ist die Kombination aus den an dieses Zertifikat `certificateChain.pem` angehängten Zertifikaten.
 - `certificateChain.pem`: Die Zertifikatskette, die das Stammzertifikat und die Zwischenzertifikate enthält.
 - `rootCertificate.pem`: Das Stammzertifikat.
 - `privateKey.pem`: Der private Schlüssel für das Endbenutzerzertifikat.
- `ComputeName`: Der Name Ihrer Rechenressource.

Beispiel

```
tlsCertificate, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

GetFleetRoleCredentials()

Ruft die Anmeldeinformationen für die Servicerolle ab, die Sie erstellen, um die Berechtigungen auf Ihre andere AWS-Services Person auszudehnen. Amazon GameLift Servers Diese Anmeldeinformationen ermöglichen es deinem Spieleserver, deine AWS Ressourcen zu nutzen.

Weitere Informationen finden Sie unter [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#).

Syntax

```
func GetFleetRoleCredentials(  
    req request.GetFleetRoleCredentialsRequest,  
) (result.GetFleetRoleCredentialsResult, error) {  
    return srv.getFleetRoleCredentials(&req)  
}
```

Parameters

[GetFleetRoleCredentialsRequest](#)

Rollenanmeldedaten, mit denen der eingeschränkte Zugriff auf deine AWS Ressourcen auf den Spieleserver ausgedehnt wird.

Rückgabewert

Gibt ein `GetFleetRoleCredentialsResult` Objekt zurück, das Folgendes enthält:

- `AssumedRoleUserArn` — Der Amazon-Ressourcenname (ARN) des Benutzers, dem die Servicerolle gehört.
- `AssumedRoleId` - Die ID des Benutzers, dem die Servicerolle gehört.
- `AccessKeyId` - Die Zugriffsschlüssel-ID zur Authentifizierung und Bereitstellung des Zugriffs auf Ihre AWS Ressourcen.
- `SecretAccessKey` - Die geheime Zugriffsschlüssel-ID für die Authentifizierung.
- `SessionToken` - Ein Token zur Identifizierung der aktuellen aktiven Sitzung, die mit Ihren AWS Ressourcen interagiert.
- `Ablauf` — Der Zeitraum, bis Ihre Sitzungsdaten ablaufen.

Beispiel

```
// form the customer credentials request  
getFleetRoleCredentialsRequest := request.NewGetFleetRoleCredentials()  
getFleetRoleCredentialsRequest.RoleArn = "arn:aws:iam::123456789012:role/service-role/  
exampleGameLiftAction"
```

```
credentials, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

Zerstören ()

Befreit das Amazon GameLift Servers Spieleserver-SDK aus dem Speicher. Es hat sich bewährt, diese Methode nach `ProcessEnding()` und vor dem Beenden des Prozesses aufzurufen. Wenn Sie eine Anywhere-Flotte verwenden und Serverprozesse nicht nach jeder Spielsitzung beenden, rufen Sie auf `Destroy()` und starten Sie dann `InitSDK()` erneut, bevor Sie benachrichtigen, Amazon GameLift Servers dass der Prozess bereit ist, eine Spielsitzung mit zu hosten. `ProcessReady()`

Syntax

```
func Destroy() error {  
    return srv.destroy()  
}
```

Rückgabewert

Gibt einen Fehler mit einer Fehlermeldung zurück, wenn die Methode fehlschlägt.

Beispiel

```
// operations to end game sessions and the server process  
defer func() {  
    err := server.ProcessEnding()  
    server.Destroy()  
    if err != nil {  
        fmt.Println("ProcessEnding() failed. Error: ", err)  
        os.Exit(-1)  
    } else {  
        os.Exit(0)  
    }  
}
```

C++ (Unreal) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen

Verwenden Sie die Amazon GameLift Servers Unreal Server SDK 5.x-Referenz, um Ihr Multiplayer-Spiel für die Verwendung mit vorzubereiten. Amazon GameLift Servers Einzelheiten zum Integrationsprozess finden Sie unter. [Füge Amazon GameLift Servers es deinem Spieleserver mit](#)

[dem Server-SDK hinzu](#) Wenn du das Amazon GameLift Servers Plugin für Unreal verwendest, siehe auch [Amazon GameLift ServersPlugin für Unreal Engine](#).

 Note

In diesem Thema wird die Amazon GameLift Servers C++-API beschrieben, die Sie beim Entwickeln für die Unreal Engine verwenden können. Insbesondere bezieht sich diese Dokumentation auf Code, den Sie mit der `-DBUILD_FOR_UNREAL=1` Option kompilieren.

C++ (Unreal) Server-SDK 5.x für Amazon GameLift Servers -- Datentypen

Verwenden Sie die Amazon GameLift Servers Unreal Server SDK 5.x-Referenz, um Ihr Multiplayer-Spiel für die Verwendung mit vorzubereiten. Amazon GameLift Servers Einzelheiten zum Integrationsprozess finden Sie unter. [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#) Wenn du das Amazon GameLift Servers Plugin für Unreal verwendest, siehe auch [Amazon GameLift ServersPlugin für Unreal Engine](#).

 Note

In diesem Thema wird die Amazon GameLift Servers C++-API beschrieben, die Sie beim Erstellen für die Unreal Engine verwenden können. Insbesondere bezieht sich diese Dokumentation auf Code, den Sie mit der `-DBUILD_FOR_UNREAL=1` Option kompilieren.

[C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)

Datentypen

- [FProcessParameter](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [FServerParameter](#)
- [FStartMatchBackfillRequest](#)
- [FPlayer](#)
- [FGameLiftDescribePlayerSessionsRequest](#)
- [FStopMatchBackfillRequest](#)

- [FAttributeWert](#)
- [FGameLiftGetFleetRoleCredentialsRequest](#)
- [FGameLiftLongOutcome](#)
- [FGameLiftStringOutcome](#)
- [FGameLiftDescribePlayerSessionsOutcome](#)
- [FGameLiftDescribePlayerSessionsResult](#)
- [FGenericErgebnis](#)
- [FGameLiftPlayerSession](#)
- [FGameLiftGetComputeCertificateOutcome](#)
- [FGameLiftGetComputeCertificateResult](#)
- [FGameLiftGetFleetRoleCredentialsOutcome](#)
- [FGetFleetRoleCredentialsResult](#)
- [FGameLiftError](#)
- [Aufzählungen](#)

FProcessParameter

Dieser Datentyp enthält den Satz von Parametern, an den Amazon GameLift Servers in a gesendet wird [ProcessReady\(\)](#).

Eigenschaften	Beschreibung
LogParameters	Ein Objekt mit Verzeichnispfaden zu Dateien, die während einer Spielsitzung generiert werden. Amazon GameLift Servers kopiert und speichert die Dateien für den future Zugriff. Typ: TArray<FString> Required: No
OnHealthCheck	Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift Servers ruft diese Funktion alle 60

Sekunden auf und wartet 60 Sekunden auf eine Antwort. Der Serverprozess kehrt zurück, TRUE wenn er fehlerfrei ist, FALSE wenn nicht. Wenn keine Antwort zurückgegeben wird, wird der Amazon GameLift Servers Serverprozess als fehlerhaft aufgezeichnet.

Diese Eigenschaft ist eine Delegatenfunktion, definiert als; DECLARE_DELEGATE_RetVal(bool, FOnHealthCheck)

Typ: FOnHealthCheck

Required: No

OnProcessTerminate

Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um das Herunterfahren des Serverprozesses zu erzwingen. Amazon GameLift Servers wartet nach dem Aufrufen dieser Funktion 5 Minuten, bis der Serverprozess heruntergefahren ist, und antwortet mit einem [ProcessEnding\(\)](#) Aufruf, bevor er den Serverprozess herunterfährt.

Typ: FSimpleDelegate

Erforderlich: Ja

OnStartGameSession

Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um eine neue Spielsitzung zu aktivieren. Amazon GameLift Servers ruft diese Funktion als Antwort auf eine Client-Anfrage auf. [CreateGameSession](#) Die Callback-Funktion übergibt ein [GameSession](#) Objekt.

Diese Eigenschaft ist eine Delegatenfunktion, definiert als `DECLARE_DELEGATE_OneParam(FOnStartGameSession, Aws::GameLift::Server::Model::GameSession);`

Typ: `FOnStartGameSession`

Erforderlich: Ja

OnUpdateGameSession

Die Callback-Funktion, die Amazon GameLift Servers aufgerufen wird, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift Servers ruft diese Funktion auf, wenn eine Match-Backfill-Anfrage bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Sie übergibt ein [GameSession](#) Objekt, ein Status-Update (`updateReason`) und die Match-Backfill-Ticket-ID.

Diese Eigenschaft ist eine Delegiertenfunktion, definiert als `DECLARE_DELEGATE_OneParam(FOnUpdateGameSession, Aws::GameLift::Server::Model::UpdateGameSession);`

Typ: `FOnUpdateGameSession`

Required: No

Port

Die Portnummer, auf der der Serverprozess auf neue Player-Verbindungen wartet. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: `int`

Erforderlich: Ja

UpdateGameSession

Dieser Datentyp aktualisiert ein Spielsitzungsobjekt, das den Grund für die Aktualisierung der Spielsitzung und die zugehörige Backfill-Ticket-ID enthält, falls Backfill zum Füllen von Spielersitzungen in der Spielsitzung verwendet wird.

Eigenschaften	Beschreibung
GameSession	Ein GameSession-Objekt. Das <code>GameSession</code> Objekt enthält Eigenschaften, die eine Spielsitzung beschreiben. Typ: <code>Aws::GameLift::Server::GameSession</code> Required: No
UpdateReason	Der Grund, warum die Spielsitzung aktualisiert wird. Typ: <code>enum class UpdateReason</code> • <code>MATCHMAKING_DATA_UPDATED</code> • <code>BACKFILL_IST FEHLGESCHLAGEN</code>

Eigenschaften	Beschreibung
	• BACKFILL_TIMED_OUT • BACKFILL_ABGESAGT Required: No
BackfillTicketId	Die ID des Backfill-Tickets, das versucht, die Spielsitzung zu aktualisieren. Typ: <code>char[]</code> Required: No

GameSession

Dieser Datentyp enthält Details einer Spielsitzung.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Eine Spielsitzung ARN hat das folgende Format: <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code>. Typ: <code>char[]</code> Required: No
Name	Eine beschreibende Bezeichnung der Spielsitzung. Typ: <code>char[]</code> Required: No
FleetId	Eine eindeutige Kennung für die Flotte, auf der die Spielsitzung läuft.

Eigenschaften	Beschreibung
	Typ: <code>char[]</code> Required: No
MaximumPlayerSessionCount	Die maximale Anzahl von Spielerverbindungen zur Spielsitzung. Typ: <code>int</code> Required: No
Port	Die Portnummer für die Spielsitzung. Um eine Verbindung zu einem Amazon GameLift Servers Spieleserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: <code>int</code> Required: No
IpAddress	Die IP-Adresse der Spielsitzung. Um eine Verbindung zu einem Amazon GameLift Servers Spieleserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: <code>char[]</code> Required: No
GameSessionData	Eine Reihe von benutzerdefinierten Spielsitzungseigenschaften, die als einzelner Zeichenfolgenwert formatiert sind. Typ: <code>char[]</code> Required: No

Eigenschaften	Beschreibung
MatchmakerData	Informationen über den Matchmaking-Prozess, der zur Erstellung der Spielsitzung verwendet wurde, in JSON-Syntax, formatiert als Zeichenfolge. Zusätzlich zur verwendeten Matchmaking-Konfiguration enthält es Daten zu allen Spielern, die dem Spiel zugewiesen wurden, einschließlich Spielerattributen und Teamzuweisungen. Typ: <code>char[]</code> Required: No
GameProperties	Eine Reihe von benutzerdefinierten Eigenschaften für eine Spielsitzung, formatiert als Schlüssel/Wert-Paare. Diese Eigenschaften werden zusammen mit der Aufforderung, eine neue Spielsitzung zu starten, übergeben. Typ: <code>GameProperty[]</code> Required: No

Eigenschaften	Beschreibung
DnsName	Die DNS-ID, die der Instanz zugewiesen ist, auf der die Spielsitzung ausgeführt wird. Werte haben das folgende Format: • TLS-fähige Flotten: <code><unique identifier>.<region identifier>.amazon gamelift.com</code> • Non-TLS-enabled Flotten: <code>ec2-<unique identifier>.compute.amazonaws.com</code> Wenn du eine Verbindung zu einer Spielsitzung herstellst, die auf einer TLS-fähigen Flotte läuft, musst du den DNS-Namen verwenden, nicht die IP-Adresse. Typ: <code>char[]</code> Required: No

FServerParameter

Informationen, die zur Aufrechterhaltung der Verbindung zwischen einem Amazon GameLift Servers Anywhere-Server und dem Amazon GameLift Servers Dienst verwendet werden. Diese Informationen werden verwendet, wenn neue Serverprozesse mit gestartet [InitSDK\(\)](#) werden. Verwenden Sie für Server, die auf Amazon GameLift Servers verwalteten EC2-Instances gehostet werden, ein leeres Objekt.

Eigenschaften	Beschreibung
webSocketUrl	Das <code>GameLiftServerSdkEndpoint</code> Amazon GameLift Servers kehrt zurück, wenn Sie RegisterCompute für eine Amazon GameLift Servers Anywhere-Rechenressource suchen.

Eigenschaften	Beschreibung
	Typ: <code>char[]</code> Erforderlich: Ja
Prozess-ID	Eine eindeutige Kennung, die für den Serverprozess registriert ist, der dein Spiel hostet. Typ: <code>char[]</code> Erforderlich: Ja
hostId	Das HostID ist die, die bei der Registrierung Ihres Computers <code>ComputeName</code> verwendet wurde. Weitere Informationen finden Sie unter RegisterCompute . Typ: <code>char[]</code> Erforderlich: Ja
FleetID	Die eindeutige Kennung der Flotte, für die der Computer registriert ist. Weitere Informationen finden Sie unter RegisterCompute . Typ: <code>char[]</code> Erforderlich: Ja
AuthToken	Das dadurch generierte Authentifizierungstoken authentifiziert Ihren Server gegenüber. Amazon GameLift Servers Amazon GameLift Servers Weitere Informationen finden Sie unter GetComputeAuthToken . Typ: <code>char[]</code> Erforderlich: Ja

FStartMatchBackfillRequest

Informationen, die zur Erstellung einer Matchmaking-Backfill-Anfrage verwendet wurden. Der Spieleserver übermittelt diese Informationen Amazon GameLift Servers in einem Anruf an.

[StartMatchBackfill\(\)](#)

Eigenschaften	Beschreibung
GameSessionArn	Eine eindeutige Kennung für die Spielsitzung. Die API-Operation GetGameSessionId gibt den Bezeichner im ARN-Format zurück. Typ: <code>char[]</code> Erforderlich: Ja
MatchmakingConfigurationArn	Eine eindeutige Kennung in Form eines ARN, die der Matchmaker für diese Anfrage verwenden kann. Der Matchmaker-ARN für die ursprüngliche Spielsitzung befindet sich im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Weitere Informationen zu Matchmaker-Daten finden Sie unter Arbeiten mit Matchmaker-Daten. Typ: <code>char[]</code> Erforderlich: Ja
Players	Ein Datensatz, der alle Spieler repräsentiert, die an der Spielsitzung teilnehmen. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Typ: <code>TArray<FPlayer></code> Erforderlich: Ja

Eigenschaften	Beschreibung
TicketId	Eine eindeutige Kennung für ein Matchmaking- oder Match-Backfill-Anfrageticket. Amazon GameLift Servers generiert einen, wenn Sie keinen Wert angeben. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzuberechnen. Typ: <code>char[]</code> Required: No

FPlayer

Dieser Datentyp steht für einen Spieler beim Matchmaking. Wenn ein Spieler eine Matchmaking-Anfrage startet, verfügt er über eine Spieler-ID, Attribute und möglicherweise Latenzdaten. Amazon GameLift Servers fügt Teaminformationen hinzu, nachdem ein Spiel gespielt wurde.

Eigenschaften	Beschreibung
LatencyInMS	Eine Reihe von Werten, ausgedrückt in Millisekunden, die die Latenz angeben, die ein Spieler erlebt, wenn er mit einem Standort verbunden ist. Wenn diese Eigenschaft verwendet wird, wird der Player nur für die aufgelisteten Standorte abgeglichen. Wenn ein Matchmaker eine Regel hat, die die Latenz der Spieler auswertet, müssen die Spieler die zu vergleichende Latenz melden. Typ: <code>TMap>FString, int32<</code> Required: No

Eigenschaften	Beschreibung
PlayerAttributes	Eine Sammlung von Schlüssel/Wert-Paaren, die Spielerinformationen zur Verwendung bei der Spielerzuweisung enthalten. Die Schlüssel der Spielerattribute müssen mit den in einem Matchmaking-Regelsatz PlayerAttributes verwendeten übereinstimmen. Weitere Informationen zu Spielerattributen finden Sie unter AttributeValue. Typ: TMap>FString, FAttributeValue< Required: No
PlayerId	Eine eindeutige Kennung für einen Spieler. Typ: std::string Required: No
Team	Der Name der Mannschaft, der der Spieler in einem Spiel zugewiesen wird. Sie definieren den Teamnamen im Matchmaking-Regelsatz. Typ: FString Required: No

FGameLiftDescribePlayerSessionsRequest

Ein Objekt, das angibt, welche Spilersitzungen abgerufen werden sollen. Der Serverprozess stellt diese Informationen mit einem [DescribePlayerSessions\(\)](#) Aufruf von bereitAmazon GameLift Servers.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielsitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist <code>FString</code>. Das <code>GameSessionID</code> ist eine benutzerdefinierte ID-Zeichenfolge oder eine Typ: <code>std::string</code> Required: No
PlayerSessionId	Die eindeutige Kennung für eine Spielsitzung. Verwenden Sie diesen Parameter, um eine einzelne spezifische Spielsitzung anzufordern. Typ: <code>FString</code> Required: No
PlayerId	Die eindeutige Kennung für einen Spieler. Verwenden Sie diesen Parameter, um alle Spielsitzungen für einen bestimmten Spieler anzufordern. Siehe Spieler generieren IDs. Typ: <code>FString</code> Required: No
PlayerSessionStatusFilter	Der Sitzungsstatus des Spielers, nach dem die Ergebnisse gefiltert werden sollen. Zu den möglichen Sitzungsstatus eines Spielers gehören: • RESERVIERT — Die Anfrage für die Spielsitzung wurde empfangen, aber der

Eigenschaften	Beschreibung
	Spieler hat keine Verbindung zum Serverprozess hergestellt oder wurde nicht validiert. • AKTIV — Der Spieler wurde durch den Serverprozess validiert und ist verbunden. • ABGESCHLOSSEN — Die Spielerverbindung wurde unterbrochen. • TIMEDOUT — Eine Anfrage für eine Spielsitzung wurde empfangen, aber der Spieler hat keine Verbindung hergestellt oder wurde nicht innerhalb des Timeoutlimits (60 Sekunden) bestätigt. Typ: FString Required: No
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite anzeigt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Spielsitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: FString Required: No
Limit	Die maximale Anzahl der auszugebenden Ergebnisse. Wenn Sie eine Spielsitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: int Required: No

FStopMatchBackfillRequest

Informationen, die verwendet werden, um eine Matchmaking-Backfill-Anfrage zu stornieren. Der Spieleserver übermittelt diese Informationen in einem Anruf an den Amazon GameLift Servers Service. [StopMatchBackfill\(\)](#)

Eigenschaften	Beschreibung
GameSessionArn	Eine eindeutige Spielsitzungs-ID der Anfrage, die storniert wird. Typ: FString Erforderlich: Ja
MatchmakingConfigurationArn	Eine eindeutige Kennung des Matchmakers, an den diese Anfrage gesendet wurde. Typ: FString Erforderlich: Ja
TicketId	Eine eindeutige Kennung des Tickets für die Backfill-Anfrage, das storniert werden soll. Typ: FString Erforderlich: Ja

FAttributeWert

Verwenden Sie diese Werte in [FPlayer](#) Attribut-Schlüssel-Wert-Paaren. Mit diesem Objekt können Sie einen Attributwert mit einem der gültigen Datentypen angeben: Zeichenfolge, Zahl, Zeichenkettenarray oder Datenmap. Jedes `AttributeValue` Objekt kann nur eine der verfügbaren Eigenschaften verwenden.

Eigenschaften	Description
AttrType	Gibt den Typ des Attributwerts an.

Eigenschaften	Description
	Typ: Ein FAttributeType Aufzählungswert . Required: No
S	Stellt einen Zeichenkettenattributwert dar. Typ: FString Required: No
N	Stellt einen numerischen Attributwert dar. Typ: double Required: No
SL	Stellt ein Array von Zeichenkettenattributwerten dar. Typ: TArray<FString> Required: No
SDM	Stellt ein Wörterbuch mit Zeichenkettenschlüsseln und Doppelwerten dar. Typ: TMap<FString, double> Required: No

FGameLiftGetFleetRoleCredentialsRequest

Dieser Datentyp stellt Rollenanmeldedaten bereit, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen auf den Spieleserver erweitern. Weitere Informationen finden Sie unter [Richten Sie eine IAM-Servicerolle ein für Amazon GameLift Servers](#).

Eigenschaften	Beschreibung
RoleArn	Der Amazon-Ressourcenname (ARN) der Servicerolle, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen erweitert. Typ: FString Required: No
RoleSessionName	Der Name der Sitzung, die die Verwendung der Rollenmeldedaten beschreibt. Typ: FString Required: No

FGameLiftLongOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: long Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: long&& Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: bool

Eigenschaften	Description
	Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "FGameLiftError" Required: No

FGameLiftStringOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: <code>FString</code> Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>FString&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "FGameLiftError"

Eigenschaften	Description
	Required: No

FGameLiftDescribePlayerSessionsOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “FGameLiftDescribePlayerSessionsResult” Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: FGameLiftDescribePlayerSessionsResult&& Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: bool Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “FGameLiftError” Required: No

FGameLiftDescribePlayerSessionsResult

Eigenschaften	Description
PlayerSessions	Typ: TArray<FGameLiftPlayerSession> Erforderlich: Ja
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite angibt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Spilersitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: FString Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: bool Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "FGameLiftError" Required: No

FGenericErgebnis

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: bool

Eigenschaften	Description
	Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "FGameLiftError" Required: No

FGameLiftPlayerSession

Eigenschaften	Description
CreationTime	Typ: long Erforderlich: Ja
FleetId	Typ: FString Erforderlich: Ja
GameSessionId	Typ: FString Erforderlich: Ja
IpAddress	Typ: FString Erforderlich: Ja
PlayerData	Typ: FString Erforderlich: Ja
PlayerId	Typ: FString Erforderlich: Ja
PlayerSessionId	Typ: FString

Eigenschaften	Description
	Erforderlich: Ja
Port	Typ: <code>int</code> Erforderlich: Ja
Status	Typ: Eine <code>PlayerSessionStatus</code> Aufzählung . Erforderlich: Ja
TerminationTime	Typ: <code>long</code> Erforderlich: Ja
DnsName	Typ: <code>FString</code> Erforderlich: Ja

FGameLiftGetComputeCertificateOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called "FGameLiftGetComputeCertificateResult" Required: No
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>FGameLiftGetComputeCertificateResult&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht.

Eigenschaften	Description
	Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "FGameLiftError" Required: No

FGameLiftGetComputeCertificateResult

Der Pfad zum TLS-Zertifikat auf Ihrem Computer und der Hostname des Computers.

Eigenschaften	Description
CertificatePath	Typ: <code>FString</code> Erforderlich: Ja
ComputeName	Typ: <code>FString</code> Erforderlich: Ja

FGameLiftGetFleetRoleCredentialsOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Description
Ergebnis	Das Ergebnis der Aktion. Typ: the section called "FGetFleetRoleCredentialsResult" Required: No

Eigenschaften	Description
ResultWithOwnership	Das Ergebnis der Aktion, in einen R-Wert umgewandelt, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: <code>FGameLiftGetFleetRoleCredentialsResult&&</code> Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: <code>bool</code> Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "FGameLiftError" Required: No

FGetFleetRoleCredentialsResult

Eigenschaften	Description
AccessKeyId	Die Zugriffsschlüssel-ID zur Authentifizierung und Bereitstellung des Zugriffs auf Ihre AWS Ressourcen. Typ: <code>FString</code> Required: No
AssumedRoleId	Die ID des Benutzers, zu dem die Servicerolle gehört. Typ: <code>FString</code> Required: No

Eigenschaften	Description
AssumedRoleUserArn	Der Amazon-Ressourcenname (ARN) des Benutzers, zu dem die Servicerolle gehört. Typ: FString Required: No
Ablauf	Die Zeitspanne, bis Ihre Sitzungsanmeldedaten ablaufen. Typ: FDateTime Required: No
SecretAccessKey	Die geheime Zugriffsschlüssel-ID für die Authentifizierung. Typ: FString Required: No
SessionToken	Ein Token zur Identifizierung der aktuellen aktiven Sitzung, die mit Ihren AWS Ressourcen interagiert. Typ: FString Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: bool Erforderlich: Ja
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called "GameLiftError" Required: No

FGameLiftError

Eigenschaften	Description
ErrorType	Der Fehlertyp. Typ: Eine GameLiftErrorType Aufzählung . Required: No
ErrorMessage	Der Name des Fehlers. Typ: std::string Required: No
ErrorMessage	Die Fehlermeldung. Typ: std::string Required: No

Aufzählungen

Die für das Server-SDK für Amazon GameLift Servers (Unreal) definierten Enums sind wie folgt definiert:

FAttributeGeben Sie ein

- NONE
- ZEICHENFOLGE
- DOPPELT
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

Zeichenkettenwert, der den Fehlertyp angibt. Gültige Werte sind:

- SERVICE_CALL_FAILED — Ein Aufruf eines Dienstes ist AWS fehlgeschlagen.

- `LOCAL_CONNECTION_FAILED` — Die lokale Verbindung zu ist fehlgeschlagen. Amazon GameLift Servers
- `NETWORK_NOT_INITIALIZED` — Das Netzwerk wurde nicht initialisiert.
- `GAMESESSION_ID_NOT_SET` — Die Spielsitzungs-ID wurde nicht festgelegt.
- `BAD_REQUEST_EXCEPTION`
- `INTERNALE_SERVICE_EXCEPTION`
- `ALREADY_INITIALIZED` — Der Amazon GameLift Servers Server oder Client wurde bereits mit `Initialize ()` initialisiert.
- `FLEET_MISMATCH` — Die Zielflotte entspricht nicht der Flotte einer `GameSession` oder `PlayerSession`.
- `GAMELIFT_CLIENT_NOT_INITIALIZED` — Der Client wurde nicht initialisiert. Amazon GameLift Servers
- `GAMELIFT_SERVER_NOT_INITIALIZED` — Amazon GameLift Servers Der Server wurde nicht initialisiert.
- `GAME_SESSION_ENDED_FAILED` — Das Server-SDK für konnte den Dienst nicht kontaktieren, um zu melden, dass die Spielsitzung beendet wurde. Amazon GameLift Servers
- `GAME_SESSION_NOT_READY` — Die Server-Spielsitzung wurde nicht aktiviert. Amazon GameLift Servers
- `GAME_SESSION_READY_FAILED` — Das Server-SDK für konnte den Dienst nicht kontaktieren, um zu melden, dass die Spielsitzung bereit ist. Amazon GameLift Servers
- `INITIALIZATION_MISMATCH` — Eine Client-Methode wurde nach `Server: :Initialize ()` aufgerufen oder umgekehrt.
- `NOT_INITIALIZED` — Der Amazon GameLift Servers Server oder Client wurde nicht mit `Initialize ()` initialisiert.
- `NO_TARGET_ALIASID_SET` — Eine Ziel-AliasID wurde nicht gesetzt.
- `NO_TARGET_FLEET_SET` — Es wurde keine Zielflotte festgelegt.
- `PROCESS_ENDING_FAILED` — Das Server-SDK für Amazon GameLift Servers konnte den Dienst nicht kontaktieren, um zu melden, dass der Prozess beendet wird.
- `PROCESS_NOT_ACTIVE` — Der Serverprozess ist noch nicht aktiv, nicht an einen gebunden und kann ihn weder annehmen noch verarbeiten. `GameSession` `PlayerSessions`
- `PROCESS_NOT_READY` — Der Serverprozess ist noch nicht bereit, aktiviert zu werden.
- `PROCESS_READY_FAILED` — Das Server-SDK für Amazon GameLift Servers konnte den Dienst nicht kontaktieren, um zu melden, dass der Prozess bereit ist.

- `SDK_VERSION_DETECTION_FAILED` — Die SDK-Versionserkennung ist fehlgeschlagen.
- `STX_CALL_FAILED` — Ein Aufruf der Server-Backend-Komponente ist fehlgeschlagen. XStx
- `STX_INITIALIZATION_FAILED` — Die Server-Backend-Komponente konnte nicht initialisiert werden. XStx
- `UNEXPECTED_PLAYER_SESSION` — Der Server hat eine nicht registrierte Spielsitzung festgestellt.
- `WEBSOCKET_CONNECT_FAILURE`
- `WEBSOCKET_CONNECT_FAILURE_FORBIDDEN`
- `WEBSOCKET_CONNECT_FAILURE_UNGÜLTIGE_URL`
- `TIMEOUT FÜR WEBSOCKET_CONNECT_FAILURE_`
- `WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE` — Behebbarer Fehler beim Senden einer Nachricht an den Dienst. GameLift WebSocket
- `WEBSOCKET_SEND_MESSAGE_FAILURE` — Fehler beim Senden einer Nachricht an GameLift den WebSocket Dienst.
- `MATCH_BACKFILL_REQUEST_VALIDATION` — Die Validierung der Anfrage ist fehlgeschlagen.
- `PLAYER_SESSION_REQUEST_VALIDATION` — Die Validierung der Anfrage ist fehlgeschlagen.

EPlayerSessionCreationPolicy

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert. Gültige Werte sind:

- `ACCEPT_ALL` – Akzeptiert alle neuen Spielsitzungen.
- `DENY_ALL` – Verwehrt neue Spielsitzungen.
- `NOT_SET` — Die Spielsitzung ist nicht so eingestellt, dass neue Spielsitzungen akzeptiert oder abgelehnt werden.

EPlayerSessionStatus

- `ACTIVE`
- `COMPLETED`
- `NOT_SET`
- `RESERVIERT`
- `TIMEOUT`

C++ (Unreal) Server-SDK 5.x für Amazon GameLift Servers -- Datentypen

Themen

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Zerstören \(\)](#)

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
FGameLiftStringOutcome GetSdkVersion();
```

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als [the section called "FGameLiftStringOutcome"](#)-Objekt zurück. Das zurückgegebene Objekt enthält die Versionsnummer

(Beispiel 5.0.0). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Initialisiert das Amazon GameLift Servers SDK für eine verwaltete EC2-Flotte. Rufen Sie diese Methode beim Start auf, bevor eine weitere Initialisierung im Zusammenhang mit erfolgt. Amazon GameLift Servers Diese Methode liest Serverparameter aus der Host-Umgebung, um die Kommunikation zwischen dem Server und dem Amazon GameLift Servers Dienst einzurichten. Sie verwendet ein Idempotenz-Token, sodass Sie diesen Aufruf sicher wiederholen können, wenn er fehlschlägt.

Syntax

```
FGameLiftGenericOutcome InitSDK()
```

Rückgabewert

Bei Erfolg wird ein `InitSdkOutcome` Objekt zurückgegeben, das angibt, dass der Serverprozess aufrufbereit ist. [ProcessReady\(\)](#)

Beispiel

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent  
to enable further communication.  
FGameLiftGenericOutcome initSdkOutcome = GameLiftSdkModule->InitSDK();
```

InitSDK()

Initialisiert das Amazon GameLift Servers SDK für eine Anywhere-Flotte oder eine verwaltete Containerflotte. Rufen Sie diese Methode beim Start auf, bevor eine weitere Initialisierung im Zusammenhang mit erfolgt. Amazon GameLift Servers Für diese Methode sind explizite Serverparameter erforderlich, um die Kommunikation zwischen dem Server und dem Amazon

GameLift Servers Dienst einzurichten. Sie verwendet ein Idempotenz-Token, sodass Sie diesen Aufruf sicher wiederholen können, wenn er fehlschlägt.

Syntax

```
FGameLiftGenericOutcome InitSDK(serverParameters)
```

Parameters

[FServerParameter](#)

Um einen Spieleserver auf einer Amazon GameLift Servers Anywhere-Flotte zu initialisieren, konstruieren Sie ein `ServerParameters` Objekt mit den folgenden Informationen:

- Die URL, mit der die WebSocket Verbindung zu Ihrem Spieleserver hergestellt wurde.
- Die ID des Prozesses, der zum Hosten deines Spieleservers verwendet wird.
- Die ID des Computers, auf dem deine Gameserver-Prozesse gehostet werden.
- Die ID der Amazon GameLift Servers Flotte, die Ihren Amazon GameLift Servers Anywhere-Computer enthält.
- Das durch den Amazon GameLift Servers Vorgang generierte Autorisierungstoken.

Rückgabewert

Bei Erfolg wird ein `InitSdkOutcome` Objekt zurückgegeben, das angibt, dass der Serverprozess aufrufbereit ist [ProcessReady\(\)](#).

Note

Wenn Aufrufe von für Spiel-Builds, die auf Anywhere-Flotten bereitgestellt werden, fehlschlagen, überprüfen Sie den `ServerSdkVersion` Parameter, der beim Erstellen der Build-Ressource verwendet wurde. `InitSDK()` Sie müssen diesen Wert explizit auf die verwendete Server-SDK-Version festlegen. Der Standardwert für diesen Parameter ist 4.x, was nicht kompatibel ist. Um dieses Problem zu beheben, erstellen Sie einen neuen Build und stellen Sie ihn in einer neuen Flotte bereit.

Beispiel

```
//Define the server parameters
```

```
FServerParameters serverParameters;  
parameters.m_authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";  
parameters.m_fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/  
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";  
parameters.m_hostId = "HardwareAnywhere";  
parameters.m_processId = "PID1234";  
parameters.m_webSocketUrl = "wss://us-west-1.api.amazongamelift.com";  
  
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent  
to enable further communication.  
FGameLiftGenericOutcome initSdkOutcome = GameLiftSdkModule->InitSDK(serverParameters);
```

ProcessReady()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess bereit ist, Spielsitzungen abzuhalten. Rufen Sie diese Methode nach dem Aufrufen auf [InitSDK\(\)](#). Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters  
&processParameters);
```

Parameters

processParameters

Ein [FProcessParameter](#) Objekt, das die folgenden Informationen über den Serverprozess übermittelt:

- Namen der im Spielservercode implementierten Callback-Methoden, die der Amazon GameLift Servers Dienst aufruft, um mit dem Serverprozess zu kommunizieren.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu beliebigen spielsitzungsspezifischen Dateien, die Amazon GameLift Servers erfassen und speichern soll.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des [ProcessReady\(\)](#)-Aufrufs und der Delegate-Funktion.

```
//Calling ProcessReady tells Amazon GameLift Servers this game server is ready to
receive incoming game sessions!
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready"));
FGameLiftGenericOutcome processReadyOutcome = GameLiftSdkModule-
>ProcessReady(*params);
```

ProcessEnding()

Benachrichtigt Amazon GameLift Servers, dass der Serverprozess beendet wird. Ruft diese Methode nach allen anderen Bereinigungsaufgaben (einschließlich des Herunterfahrens der aktiven Spielsitzung) und vor dem Beenden des Vorgangs auf. Je nach Ergebnis von wird der `ProcessEnding()` Prozess mit Erfolg (0) oder Fehler (-1) beendet und generiert ein Flottenereignis. Wenn der Prozess mit einem Fehler beendet wird, ist `SERVER_PROCESS_TERMINATED_UNHEALTHY` das generierte Flottenereignis ().

Syntax

```
FGameLiftGenericOutcome ProcessEnding()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
//OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback before
shutting down an instance hosting this game server.
//It gives this game server a chance to save its state, communicate with services,
etc., before being shut down.
//In this case, we simply tell Amazon GameLift Servers we are indeed going to shutdown.
params->OnTerminate.BindLambda( [=]() {
    UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
    GameLiftSdkModule->ProcessEnding();
});
```

ActivateGameSession()

Meldet Amazon GameLift Servers, dass der Serverprozess eine Spielsitzung aktiviert hat und nun bereit ist, Spielerverbindungen zu empfangen. Diese Aktion sollte als Teil der `onStartGameSession()` Callback-Funktion aufgerufen werden, nachdem alle Spielsitzungen initialisiert wurden.

Syntax

```
FGameLiftGenericOutcome ActivateGameSession()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel zeigt, dass es als Teil der `onStartGameSession()` Delegate-Funktion `ActivateGameSession()` aufgerufen wird.

```
//When a game session is created, Amazon GameLift Servers sends an activation request
//to the game server and passes along the game session object containing game properties
//and other settings.
//Here is where a game server should take action based on the game session object.
//Once the game server is ready to receive incoming player connections, it should
//invoke GameLiftServerAPI.ActivateGameSession()
auto onGameSession = [](Aws::GameLift::Server::Model::GameSession gameSession)
{
    FString gameId = FString(gameSession.GetGameSessionId());
    UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"), *gameId);
    GameLiftSdkModule->ActivateGameSession();
};
```

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielsitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt.

Syntax

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy policy)
```

Parameters

playerCreationSessionRichtlinie

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
FGameLiftGenericOutcome outcome = GameLiftSdkModule->UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::EPlayerSessionCreationPolicy::ACCEPT_A
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die vom aktiven Serverprozess gehostet wird.

Bei Prozessen im Leerlauf, die während einer Spielsitzung nicht aktiviert wurden, gibt der Aufruf a [the section called “FGameLiftError”](#) zurück.

Syntax

```
FGameLiftStringOutcome GetGameSessionId()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als [the section called "FGameLiftStringOutcome"](#)-Objekt zurück. Wenn das nicht erfolgreich ist, wird eine Fehlermeldung zurückgegeben.“

Bei inaktiven Prozessen, die nicht während einer Spielsitzung aktiviert werden, gibt der Aufruf `Success = True` und `GameSessionId = zurück""`.

Beispiel

```
//When a game session is created, Amazon GameLift Servers sends an activation request
//to the game server and passes along the game session object containing game properties
//and other settings.
//Here is where a game server should take action based on the game session object.
//Once the game server is ready to receive incoming player connections, it should
//invoke GameLiftServerAPI.ActivateGameSession()
auto onGameSession = [](Aws::GameLift::Server::Model::GameSession gameSession)
{
    FString gameSessionId = FString(gameSession.GetGameSessionId());
    UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"), *gameSessionId);
    GameLiftSdkModule->ActivateGameSession();
};
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess ergreift eine Aktion, nachdem er einen `onProcessTerminate()` Rückruf von Amazon GameLift Servers erhalten hat. Amazon GameLift Servers ruft `onProcessTerminate()` aus den folgenden Gründen auf:

- Wenn der Serverprozess einen schlechten Zustand gemeldet hat oder nicht darauf reagiert hat Amazon GameLift Servers.
- Wenn die Instanz während eines Scale-Down-Ereignisses beendet wird.
- [Wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.](#)

Syntax

```
AwsDateTimeOutcome GetTerminationTime()
```

Rückgabewert

Bei Erfolg wird die Kündigungszeit als Objekt `AwsDateTimeOutcome` zurückgegeben. Der Wert ist die Endzeit, ausgedrückt in verstrichenen Ticks, die seitdem vergangen sind. `0001 00:00:00` Beispielsweise entspricht der Wert für Datum und Uhrzeit den `2020-09-13 12:26:40 -000Z` Ticks. `637355968000000000` Wenn keine Kündigungszeit verfügbar ist, wird eine Fehlermeldung zurückgegeben.

Wenn der Prozess keinen `ProcessParameters.OnProcessTerminate()` Rückruf erhalten hat, wird eine Fehlermeldung zurückgegeben. Weitere Hinweise zum Herunterfahren eines Serverprozesses finden Sie unter. [Reagieren Sie auf eine Benachrichtigung zum Herunterfahren des Serverprozesses](#)

Beispiel

```
AwsDateTimeOutcome TermTimeOutcome = GameLiftSdkModule->GetTerminationTime();
```

AcceptPlayerSession()

Benachrichtigt Amazon GameLift Servers, dass ein Spieler mit der angegebenen Spilersitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Bestätigung benötigt. Amazon GameLift Servers überprüft, ob die Player-Sitzungs-ID gültig ist. Nachdem die Spilersitzung bestätigt wurde, wird der Status des Spieler-Slots von RESERVIERT auf AKTIV Amazon GameLift Servers geändert.

Syntax

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

Parameters

playerSessionId

Eindeutige ID Amazon GameLift Servers, die ausgestellt wird, wenn eine neue Spilersitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird eine Verbindungsanforderung behandelt, die das Überprüfen und Ablehnen einer ungültigen Spielsitzung beinhaltet. IDs

```
bool GameLiftManager::AcceptPlayerSession(const FString& playerIdSessionId, const
    FString& playerId)
{
    #if WITH_GAMELIFT
        UE_LOG(GameServerLog, Log, TEXT("Accepting GameLift PlayerSession: %s . PlayerId:
        %s"), *playerSessionId, *playerId);
        FString gsId = GetCurrentGameSessionId();
        if (gsId.IsEmpty()) {
            UE_LOG(GameServerLog, Log, TEXT("No GameLift GameSessionId. Returning early!"));
            return false;
        }

        if (!GameLiftSdkModule->AcceptPlayerSession(playerSessionId).IsSuccess()) {
            UE_LOG(GameServerLog, Log, TEXT("PlayerSession not Accepted.));
            return false;
        }

        // Add PlayerSession from internal data structures keeping track of connected players
        connectedPlayerSessionIds.Add(playerSessionId);
        idToPlayerSessionMap.Add(playerSessionId, PlayerSession{ playerId,
        playerIdSessionId });
        return true;
    #else
        return false;
    #endif
}
```

RemovePlayerSession()

Benachrichtigt Amazon GameLift Servers, dass ein Spieler die Verbindung zum Serverprozess getrennt hat. Amazon GameLift Servers ändert daraufhin den Player-Slot auf „Verfügbar“.

Syntax

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerId)
```

Parameters

playerSessionId

Eindeutige ID Amazon GameLift Servers, die ausgestellt wird, wenn eine neue Spielsitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
bool GameLiftManager::RemovePlayerSession(const FString& playerId)
{
    #if WITH_GAMELIFT
        UE_LOG(GameServerLog, Log, TEXT("Removing GameLift PlayerSession: %s"),
            *playerSessionId);

        if (!GameLiftSdkModule->RemovePlayerSession(playerSessionId).IsSuccess()) {
            UE_LOG(GameServerLog, Log, TEXT("PlayerSession Removal Failed"));
            return false;
        }

        // Remove PlayerSession from internal data structures that are keeping track of
        // connected players
        connectedPlayerSessionIds.Remove(playerSessionId);
        idToPlayerSessionMap.Remove(playerSessionId);

        // end the session if there are no more players connected
        if (connectedPlayerSessionIds.Num() == 0) {
            EndSession();
        }

        return true;
    #else
        return false;
    #endif
}
```

```
#endif  
}
```

DescribePlayerSessions()

Ruft Spielersitzungsdaten ab, einschließlich Einstellungen, Sitzungsmetadaten und Spielerdaten. Verwenden Sie diese Methode, um Informationen zu folgenden Themen abzurufen:

- Eine Einzelspieler-Sitzung
- Alle Spielersitzungen in einer Spielsitzung
- Alle Spielersitzungen, die mit einer einzigen Spieler-ID verknüpft sind

Syntax

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const  
    FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

Parameters

FGameLiftDescribePlayerSessionsRequest

Ein [the section called “FGameLiftDescribePlayerSessionsRequest”](#) Objekt, das beschreibt, welche Spielersitzungen abgerufen werden sollen.

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein [the section called “FGameLiftDescribePlayerSessionsOutcome”](#)-Objekt mit einer Menge von Spielersitzungsobjekten zurück, die den Anforderungsparametern entsprechen.

Beispiel

In diesem Beispiel werden alle Spielersitzungen abgefragt, die aktiv mit einer bestimmten Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen NextToken und ihn auf 10 setzen, werden die Sitzungsdatensätze der ersten 10 Spieler Amazon GameLift Servers zurückgegeben, die der Anfrage entsprechen.

```
void GameLiftManager::DescribePlayerSessions()  
{
```

```

#if WITH_GAMELIFT
FString localPlayerSessions;
for (auto& psId : connectedPlayerSessionIds)
{
    PlayerSession ps = idToPlayerSessionMap[psId];
    localPlayerSessions += FString::Printf(TEXT("%s : %s ; "), *(ps.playerSessionId),
*(ps.playerId));
}
UE_LOG(GameServerLog, Log, TEXT("LocalPlayerSessions: %s"), *localPlayerSessions);

UE_LOG(GameServerLog, Log, TEXT("Describing PlayerSessions in this GameSession"));
FGameLiftDescribePlayerSessionsRequest request;
request.m_gameSessionId = GetCurrentGameSessionId();

FGameLiftDescribePlayerSessionsOutcome outcome = GameLiftSdkModule-
>DescribePlayerSessions(request);
LogDescribePlayerSessionsOutcome(outcome);
#endif
}

```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Weitere Informationen finden Sie unter [FlexMatchBackfill-Funktion](#).

Diese Aktion ist asynchron. Wenn neue Spieler gematcht werden, Amazon GameLift Servers liefert es mithilfe der Callback-Funktion aktualisierte Matchmaker-Daten. OnUpdateGameSession()

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst [StopMatchBackfill\(\)](#) auf, um die ursprüngliche Anforderung abzuberechnen.

Syntax

```

FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest
&startBackfillRequest);

```

Parameters

[FStartMatchBackfillRequest](#)

Ein StartMatchBackfillRequest Objekt, das die folgenden Informationen übermittelt:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Information ist optional. Wenn keine ID angegeben wird, Amazon GameLift Servers wird eine generiert.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert ist in den Matchmaker-Daten der Spielsitzung enthalten.
- Die ID der Spielsitzung, die aufgefüllt werden soll.
- Die verfügbaren Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Rückgabewert

Gibt ein `StartMatchBackfillOutcome` Objekt mit der ID des Match-Backfill-Tickets zurück oder schlägt fehl und es wird eine Fehlermeldung angezeigt.

Beispiel

```
FGameLiftStringOutcome FGameLiftServerSDKModule::StartMatchBackfill(const
    FStartMatchBackfillRequest& request)
{
    #if WITH_GAMELIFT
    Aws::GameLift::Server::Model::StartMatchBackfillRequest sdkRequest;
    sdkRequest.SetTicketId(TCHAR_TO_UTF8(*request.m_ticketId));
    sdkRequest.SetGameSessionArn(TCHAR_TO_UTF8(*request.m_gameSessionArn));

    sdkRequest.SetMatchmakingConfigurationArn(TCHAR_TO_UTF8(*request.m_matchmakingConfigurationArn));
    for (auto player : request.m_players) {
        Aws::GameLift::Server::Model::Player sdkPlayer;
        sdkPlayer.SetPlayerId(TCHAR_TO_UTF8(*player.m_playerId));
        sdkPlayer.SetTeam(TCHAR_TO_UTF8(*player.m_team));
        for (auto entry : player.m_latencyInMs) {
            sdkPlayer.WithLatencyMs(TCHAR_TO_UTF8(*entry.Key), entry.Value);
        }

        std::map<std::string, Aws::GameLift::Server::Model::AttributeValue>
        sdkAttributeMap;
        for (auto attributeEntry : player.m_playerAttributes) {
            FAttributeValue value = attributeEntry.Value;
            Aws::GameLift::Server::Model::AttributeValue attribute;
            switch (value.m_type) {
                case FAttributeType::STRING:
                    attribute =
            Aws::GameLift::Server::Model::AttributeValue(TCHAR_TO_UTF8(*value.m_S));
```

```

        break;
        case FAttributeType::DOUBLE:
            attribute = Aws::GameLift::Server::Model::AttributeValue(value.m_N);
            break;
        case FAttributeType::STRING_LIST:
            attribute =
Aws::GameLift::Server::Model::AttributeValue::ConstructStringList();
            for (auto sl : value.m_SL) {
                attribute.AddString(TCHAR_TO_UTF8(*sl));
            };
            break;
        case FAttributeType::STRING_DOUBLE_MAP:
            attribute =
Aws::GameLift::Server::Model::AttributeValue::ConstructStringDoubleMap();
            for (auto sdm : value.m_SDM) {
                attribute.AddStringAndDouble(TCHAR_TO_UTF8(*sdm.Key), sdm.Value);
            };
            break;
    }
    sdkPlayer.WithPlayerAttribute((TCHAR_TO_UTF8(*attributeEntry.Key)), attribute);
}
sdkRequest.AddPlayer(sdkPlayer);
}
auto outcome = Aws::GameLift::Server::StartMatchBackfill(sdkRequest);
if (outcome.IsSuccess()) {
    return FGameLiftStringOutcome(outcome.GetResult().GetTicketId());
}
else {
    return FGameLiftStringOutcome(FGameLiftError(outcome.GetError()));
}
#else
return FGameLiftStringOutcome("");
#endif
}

```

StopMatchBackfill()

Bricht eine aktive Match-Backfill-Anfrage ab. Weitere Informationen finden Sie unter [FlexMatchBackfill-Funktion](#).

Syntax

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest  
&stopBackfillRequest);
```

Parameters

[FStopMatchBackfillRequest](#)

Ein `StopMatchBackfillRequest` Objekt, das das zu stornierende Matchmaking-Ticket identifiziert:

- Die der Backfill-Anfrage zugewiesene Ticket-ID.
- Der Matchmaker, an den die Backfill-Anfrage gesendet wurde.
- Die Spielsitzung, die mit der Backfill-Anfrage verknüpft ist.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
FGameLiftGenericOutcome FGameLiftServerSDKModule::StopMatchBackfill(const  
    FStopMatchBackfillRequest& request)  
{  
    #if WITH_GAMELIFT  
        Aws::GameLift::Server::Model::StopMatchBackfillRequest sdkRequest;  
        sdkRequest.SetTicketId(TCHAR_TO_UTF8(*request.m_ticketId));  
        sdkRequest.SetGameSessionArn(TCHAR_TO_UTF8(*request.m_gameSessionArn));  
  
        sdkRequest.SetMatchmakingConfigurationArn(TCHAR_TO_UTF8(*request.m_matchmakingConfigurationArn));  
        auto outcome = Aws::GameLift::Server::StopMatchBackfill(sdkRequest);  
        if (outcome.IsSuccess()) {  
            return FGameLiftGenericOutcome(nullptr);  
        }  
        else {  
            return FGameLiftGenericOutcome(FGameLiftError(outcome.GetError()));  
        }  
    #else  
        return FGameLiftGenericOutcome(nullptr);  
    #endif  
}
```

```
}
```

GetComputeCertificate()

Ruft den Pfad zu dem TLS-Zertifikat ab, mit dem die Netzwerkverbindung zwischen Ihrer Amazon GameLift Servers Anywhere-Rechenressource und verschlüsselt wurde. Amazon GameLift Servers Sie können den Zertifikatspfad verwenden, wenn Sie Ihr Computergerät bei einer Amazon GameLift Servers Anywhere-Flotte registrieren. Weitere Informationen finden Sie unter [RegisterCompute](#).

Syntax

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
```

Rückgabewert

Gibt ein `GetComputeCertificateResponse` Objekt zurück, das Folgendes enthält:

- **CertificatePath:** Der Pfad zum TLS-Zertifikat auf Ihrer Rechenressource.
- **HostName:** Der Hostname Ihrer Rechenressource.

Beispiel

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
{
    #if WITH_GAMELIFT
    auto outcome = Aws::GameLift::Server::GetComputeCertificate();
    if (outcome.IsSuccess()) {
        auto& outres = outcome.GetResult();
        FGameLiftGetComputeCertificateResult result;
        result.m_certificate_path = UTF8_T0_TCHAR(outres.GetCertificatePath());
        result.m_computeName = UTF8_T0_TCHAR(outres.GetComputeName());
        return FGameLiftGetComputeCertificateOutcome(result);
    }
    else {
        return FGameLiftGetComputeCertificateOutcome(FGameLiftError(outcome.GetError()));
    }
    #else
    return FGameLiftGetComputeCertificateOutcome(FGameLiftGetComputeCertificateResult());
    #endif
}
```


GetFleetRoleCredentials()

Ruft Anmeldeinformationen für IAM-Rollen ab, die Amazon GameLift Servers zur Interaktion mit anderen berechtigen. AWS-Services Weitere Informationen finden Sie unter [Connect deinen Amazon GameLift Servers gehosteten Spieleserver mit anderen AWS Ressourcen](#).

Syntax

```
FGameLiftGetFleetRoleCredentialsOutcome  
FGameLiftServerSDKModule::GetFleetRoleCredentials(const  
FGameLiftGetFleetRoleCredentialsRequest &request)
```

Parameters

[FGameLiftGetFleetRoleCredentialsRequest](#)

Rückgabewert

Gibt ein [the section called “FGameLiftGetFleetRoleCredentialsOutcome”](#)-Objekt zurück.

Beispiel

```
FGameLiftGetFleetRoleCredentialsOutcome  
FGameLiftServerSDKModule::GetFleetRoleCredentials(const  
FGameLiftGetFleetRoleCredentialsRequest &request)  
{  
    #if WITH_GAMELIFT  
    Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest sdkRequest;  
    sdkRequest.SetRoleArn(TCHAR_TO_UTF8(*request.m_roleArn));  
    sdkRequest.SetRoleSessionName(TCHAR_TO_UTF8(*request.m_roleSessionName));  
  
    auto outcome = Aws::GameLift::Server::GetFleetRoleCredentials(sdkRequest);  
  
    if (outcome.IsSuccess()) {  
        auto& outres = outcome.GetResult();  
        FGameLiftGetFleetRoleCredentialsResult result;  
        result.m_assumedUserRoleArn = UTF8_TO_TCHAR(outres.GetAssumedUserRoleArn());  
        result.m_assumedRoleId = UTF8_TO_TCHAR(outres.GetAssumedRoleId());  
        result.m_accessKeyId = UTF8_TO_TCHAR(outres.GetAccessKeyId());  
        result.m_secretAccessKey = UTF8_TO_TCHAR(outres.GetSecretAccessKey());  
        result.m_sessionToken = UTF8_TO_TCHAR(outres.GetSessionToken());  
        result.m_expiration = FDateTime::FromUnixTimestamp(outres.GetExpiration());  
        return FGameLiftGetFleetRoleCredentialsOutcome(result);  
    }  
}
```

```
}  
else {  
    return FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftError(outcome.GetError()));  
}  
#else  
return  
FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftGetFleetRoleCredentialsResult());  
#endif  
}
```

Zerstören ()

Befreit das Amazon GameLift Servers Spieleserver-SDK aus dem Speicher. Es hat sich bewährt, diese Methode nach `ProcessEnding()` und vor dem Beenden des Prozesses aufzurufen. Wenn Sie eine Anywhere-Flotte verwenden und Serverprozesse nicht nach jeder Spielsitzung beenden, rufen Sie auf `Destroy()` und starten Sie dann `InitSDK()` erneut, bevor Sie benachrichtigen, Amazon GameLift Servers dass der Prozess bereit ist, eine Spielsitzung mit zu hosten. `ProcessReady()`

Syntax

```
FGameLiftGenericOutcome Destroy();
```

Parameters

Es gibt keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
// First call ProcessEnding()  
FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule->ProcessEnding();  
  
// Then call Destroy() to free the SDK from memory  
FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();  
  
// Exit the process with success or failure  
if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess())
```

```
{
    UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
}
else {
    if (!processEndingOutcome.IsSuccess()) {
        const FGameLiftError& error = processEndingOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("ProcessEnding() failed. Error: %s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
            *error.m_errorMessage);
    }
    if (!destroyOutcome.IsSuccess()) {
        const FGameLiftError& error = destroyOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("Destroy() failed. Error: %s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
            *error.m_errorMessage);
    }
}
```

Server-SDK für Amazon GameLift Servers Version 4 und früher

Diese Referenz dokumentiert das Server-SDK für Amazon GameLift Servers Version 4.x und früher. Das Server-SDK bietet Kernfunktionen, die Ihre Spielservers für die Kommunikation mit dem Amazon GameLift Servers Dienst verwenden. Ihr Spielservers erhält beispielsweise vom Dienst Aufforderungen, Spielsitzungen zu starten und zu beenden, und er informiert den Dienst regelmäßig über den Status der Spielsitzungen. Integrieren Sie Ihre Spielservers in das Server-SDK, bevor Sie sie zum Hosten einsetzen.

Verwenden Sie diese Server-SDK-Referenz, um Ihre benutzerdefinierten Multiplayer-Spielservers zum Hosten zu integrieren Amazon GameLift Servers. Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spielservers mit dem Server-SDK hinzu](#).

Die SDK-Version 4.0.2 können Sie von den [offiziellen GitHub Versionen](#) herunterladen. Das komplette SDK-Paket einschließlich GameLiftLocal.jar ist im Artefakt GameLift-CSharp-ServerSDK-4.0.2.zip verfügbar.

Die neueste Hauptversion des Server-SDK für ist 5.x. Amazon GameLift Servers Für die folgenden Hosting-Funktionen sind Updates auf Version 5.x erforderlich:

- Amazon GameLift ServersIrgendwo
- Amazon GameLift Serversverwaltete Container

- Amazon GameLift ServersPlugin für Unreal Engine und Unity

Themen

- [C++-Server-SDK für Amazon GameLift Servers 4.x -- Aktionen](#)
- [C#-Server-SDK für Amazon GameLift Servers 4.x — Aktionen](#)
- [Server-SDK \(Unreal\) für Amazon GameLift Servers -- Aktionen](#)

C++-Server-SDK für Amazon GameLift Servers 4.x -- Aktionen

Verwenden Sie die Server-SDK-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrierenAmazon GameLift Servers. Hinweise zum Integrationsprozess finden Sie unter[Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Note

Diese Referenz bezieht sich auf eine frühere Version des Server-SDK fürAmazon GameLift Servers. Informationen zur neuesten Version finden Sie unter [C++-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#).

C++-Server-SDK für Amazon GameLift Servers 4.x -- Datentypen

Verwenden Sie die Server-SDK-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrierenAmazon GameLift Servers. Hinweise zum Integrationsprozess finden Sie unter[Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Note

Diese Referenz bezieht sich auf eine frühere Version des Server-SDK fürAmazon GameLift Servers. Informationen zur neuesten Version finden Sie unter [C++-Server-SDK 5.x für Amazon GameLift Servers -- Datentypen](#).

Diese API wird in `GameLiftServerAPI.h`, `LogParameters.h` und `ProcessParameters.h` definiert.

[C++-Server-SDK für Amazon GameLift Servers 4.x -- Aktionen](#)

DescribePlayerSessionsRequest

Dieser Datentyp wird verwendet, um anzugeben, welche Spielsitzung(en) abgerufen werden sollen. Sie können ihn wie folgt verwenden:

- Geben Sie a `PlayerSessionId` an, um eine bestimmte Spielsitzung anzufordern.
- Geben Sie a `GameSessionId` an, um alle Spielsitzungen in der angegebenen Spielsitzung anzufordern.
- Geben Sie a `PlayerId` an, um alle Spielsitzungen für den angegebenen Spieler anzufordern.

Verwenden Sie für große Sammlungen von Spielsitzungen die Paginierungsparameter, um Ergebnisse in aufeinander folgenden Blöcken abzurufen.

Inhalt

GameSessionId

Eindeutiger Bezeichner für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielsitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist wie folgt: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. Der Wert von `<ID-String>` ist entweder eine benutzerdefinierte ID-Zeichenfolge (sofern bei der Erstellung der Spielsitzung angegeben) oder eine generierte Zeichenfolge.

Typ: Zeichenfolge

Erforderlich: Nein

Limit

Maximale Anzahl der zurückzugebenden Ergebnisse. Verwenden Sie diesen Parameter mit `NextToken`, um Ergebnisse in Form einer Reihe von aufeinanderfolgenden Seiten zu erhalten. Wenn eine Spielsitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Ganzzahl

Erforderlich: Nein

NextToken

Token, der den Beginn der nächsten Seite mit Ergebnissen anzeigt. Verwenden Sie den Token, der bei einem früheren Aufruf dieser Aktion zurückgegeben wurde. Um den Beginn des

Ergebnissatzes anzugeben, geben Sie keinen Wert an. Wenn eine Spiellersitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Zeichenfolge

Erforderlich: Nein

PlayerId

Eindeutiger Bezeichner für den Spieler. Spieler IDs werden vom Entwickler definiert. Siehe [Spieler generieren IDs](#).

Typ: Zeichenfolge

Erforderlich: Nein

PlayerSessionId

Eindeutiger Bezeichner für eine Spiellersitzung.

Typ: Zeichenfolge

Erforderlich: Nein

PlayerSessionStatusFilter

Status der Spiellersitzung für die Filterung der Ergebnisse. Mögliche Spiellersitzungsstatus sind u. a.:

- RESERVIERT — Die Anfrage für die Spiellersitzung wurde empfangen, aber der Spieler hat sich noch nicht mit dem Serverprozess verbunden, der bestätigt and/or wurde.
- ACTIVE – Der Spieler wurden vom Serverprozess validiert und ist derzeit verbunden.
- COMPLETED – Die Spielerverbindung wurde beendet.
- TIMEDOUT — Eine Anfrage für eine Spiellersitzung wurde empfangen, aber der Spieler hat keine Verbindung hergestellt und das Timeout-Limit (60 Sekunden) and/or wurde nicht bestätigt.

Typ: Zeichenfolge

Erforderlich: Nein

LogParameters

Dieser Datentyp wird verwendet, um zu identifizieren, welche Dateien während einer Sitzung generiert werden, die Sie in Amazon GameLift Servers hochladen und nach Beendigung der

Spielsitzung speichern möchten. Diese Informationen werden dem Amazon GameLift Servers-Service in einem [ProcessReady\(\)](#)-Aufruf übermittelt.

Inhalt

logPaths

Verzeichnispfade zu Spielserver-Protokolldateien, die Amazon GameLift Servers für den späteren Zugriff speichern soll. Diese Dateien werden während jeder Spielsitzung generiert. Dateipfade und Namen sind in Ihrem Spielserver definiert und im Root-Spiel-Build-Verzeichnis gespeichert. Die Protokollpfade müssen absolut sein. Zum Beispiel: Wenn Ihr Spiel-Build Spielsitzungsprotokolle in einem Pfad wie `MyGame\sessionlogs\` speichert, ist der Protokollpfad `c:\game\MyGame\sessionLogs` (auf einer Windows-Instance) oder `/local/game/MyGame/sessionLogs` (auf einer Linux-Instance).

Typ: `std::vector<std::string>`

Erforderlich: Nein

ProcessParameters

Dieser Datentyp enthält den Parametersatz, der in einem [ProcessReady\(\)](#)-Aufruf an den Amazon GameLift Servers-Service gesendet wurde.

Inhalt

port

Nummer des Ports, den der Serverprozess für neue Spielerverbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: Ganzzahl

Erforderlich: Ja

logParameters

Objekt mit einer Liste der Verzeichnispfade zu den Spielsitzungsprotokolldateien.

Typ: `Aws::GameLift::Server::` [LogParameters](#)

Erforderlich: Nein

onStartGameSitzung

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um eine neue Spielsitzung zu aktivieren. Amazon GameLift Servers ruft diese Funktion als Antwort auf die Client-Anfrage auf. [CreateGameSession](#) Die Callback-Funktion übergibt ein [GameSession](#) Objekt (definiert in der Amazon GameLift ServersService API-Referenz).

```
Typ: const std::function<void(Aws::GameLift::Model::GameSession)>  
onStartGameSession
```

Erforderlich: Ja

onProcessTerminate

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion wartet Amazon GameLift Servers fünf Minuten auf die Beendigung des Serverprozesses und antwortet mit einem [ProcessEnding\(\)](#)-Aufruf. Wird keine Antwort empfangen, wird der Server-Prozess beendet.

```
Typ: std::function<void()> onProcessTerminate
```

Erforderlich: Nein

onHealthCheck

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift Servers ruft diese Funktion alle 60 Sekunden auf. Nach dem Aufruf dieser Funktion wartet Amazon GameLift Servers 60 Sekunden auf eine Antwort; wenn keine Antwort eingeht, wird der Serverprozess als fehlerhaft notiert.

```
Typ: std::function<bool()> onHealthCheck
```

Erforderlich: Nein

onUpdateGameSitzung

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift Servers ruft diese Funktion auf, wenn eine [Match-Backfill-Anfrage](#) bearbeitet wurde, um

aktualisierte Matchmaker-Daten bereitzustellen. Sie übergibt ein [GameSession](#)-Objekt, ein Status-Update (updateReason) und die Match-Backfill-Ticket-ID.

Typ: `std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)>`
`onUpdateGameSession`

Erforderlich: Nein

StartMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung zu versenden. Die Informationen werden dem Amazon GameLift Servers-Service in einem [StartMatchBackfill\(\)](#)-Aufruf übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner für die Spielsitzung. Die API-Aktion [GetGameSessionId\(\)](#) gibt den Bezeichner im ARN-Format zurück.

Typ: Zeichenfolge

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner in Form eines ARN für den Matchmaker, der für diese Anfrage verwendet werden soll. Um den Matchmaker zu finden, der für die Erstellung der ursprünglichen Spielsitzung verwendet wurde, suchen Sie im Spielsitzungsobjekt in der Dateneigenschaft Matchmaker. Erfahren Sie mehr über Matchmaker-Daten in [Word mit Matchmaker-Daten](#).

Typ: Zeichenfolge

Erforderlich: Ja

Players

Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Eine Beschreibung des Player-Objektformats finden Sie im Amazon GameLift Servers-API-Referenzhandbuch. Spielerattribute und Teamzuweisungen finden Sie im

Objekt „Spielsitzung“ in der Eigenschaft „Matchmaker-Daten“. IDs Wenn der Matchmaker die Latenz nutzt, erfassen Sie die aktualisierte Latenz für die aktuelle Region und nehmen Sie sie in die Daten jedes Spielers auf.

Typ: std::vectorhttps://docs.aws.amazon.com/gamelift/latest/apireference/API_Player.html
<player>

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner für ein Matchmaking- oder Match-Backfill-Request-Ticket. Wenn hier kein Wert angegeben wird, Amazon GameLift Servers wird einer in Form einer UUID generiert. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abubrechen.

Typ: Zeichenfolge

Erforderlich: Nein

StopMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung abubrechen. Die Informationen werden dem Amazon GameLift Servers-Service in einem [StopMatchBackfill\(\)](#)-Aufruf übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner der Spielsitzung für die Abbruchanfrage.

Typ: Zeichenfolge

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner des Matchmakers, an den diese Anfrage gesendet wurde.

Typ: Zeichenfolge

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner des abzubrechenden Tickets für die Backfill-Anforderung.

Typ: Zeichenfolge

Erforderlich: Ja

C++-Server-SDK für Amazon GameLift Servers 4.x -- Datentypen

Themen

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [Zerstören \(\)](#)

AcceptPlayerSession()

Informiert den Amazon GameLift Servers Dienst darüber, dass ein Spieler mit der angegebenen Spielersitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Bestätigung benötigt.

Amazon GameLift Servers überprüft, ob die Spielsitzungs-ID gültig ist, d. h. ob die Spieler-ID einen Spielerplatz in der Spielsitzung reserviert hat. Nach der Bestätigung ändert Amazon GameLift Servers den Status des Spielerplatzes von RESERVED nach ACTIVE.

Syntax

```
GenericOutcome AcceptPlayerSession(const std::string& playerSessionId);
```

Parameters

playerSessionId

Eindeutige ID, die vom Amazon GameLift Servers Dienst als Antwort auf einen Aufruf der SDK-API-Aktion ausgegeben wird AWS . Amazon GameLift Servers [CreatePlayerSession](#) Der Spielclient verweist auf diese ID, wenn er eine Verbindung zum Serverprozess herstellt.

Typ: `std::string`

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel zeigt eine Funktion zur Bearbeitung einer Verbindungsanfrage, einschließlich der Validierung und Ablehnung einer ungültigen Spielsitzung. IDs

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
playerSessionId){
    Aws::GameLift::GenericOutcome connectOutcome =
        Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
    if(connectOutcome.IsSuccess())
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
```

```
{  
    connection.Reject(connectOutcome.GetError().GetMessage());  
}  
}
```

ActivateGameSession()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess eine Sitzungs-ID für das Spiel gestartet hat und nun Verbindungen annehmen kann. Diese Aktion muss im Rahmen der `onStartGameSession()`-Callback-Funktion aufgerufen werden, nachdem die Initialisierung der Spielsitzung vollständig abgeschlossen ist.

Syntax

```
GenericOutcome ActivateGameSession();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird `ActivateGameSession()` als Teil der `onStartGameSession()`-Callback-Funktion aufgerufen.

```
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)  
{  
    // game-specific tasks when starting a new game session, such as loading map  
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession();  
}
```

DescribePlayerSessions()

Ruft Sitzungsdaten der Spieler, einschließlich Einstellungen, Metadaten zu der Sitzung und Spielerdaten ab. Verwenden Sie diese Aktion, um für eine einzelne Spielsitzung, für alle

Spielersitzungen in einer Spielsitzung oder für alle Spielersitzungen zu einer einzelnen Spieler-ID abzurufen.

Syntax

```
DescribePlayerSessionsOutcome DescribePlayerSessions (  
    const Aws::GameLift::Server::Model::DescribePlayerSessionsRequest  
    &describePlayerSessionsRequest);
```

Parameters

describePlayerSessionsAnfrage

Ein [DescribePlayerSessionsRequest](#)-Objekt, mit dem beschrieben wird, welche Spielersitzungen abgerufen werden sollen.

Erforderlich: Ja

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein `DescribePlayerSessionsOutcome`-Objekt mit einer Menge von Spielersitzungsobjekten zurück, die den Anforderungsparametern entsprechen. Player-Sitzungsobjekte haben eine Struktur, die mit dem AWS Amazon GameLift Servers [PlayerSession](#) SDK-API-Datentyp identisch ist.

Beispiel

Dieses Beispiel zeigt eine Anforderung für alle Spielersitzungen, die derzeit aktiv mit einer angegebenen Spielsitzung verbunden sind. Wenn `NextToken` ausgelassen und der `Limit`-Wert auf 10 gesetzt wird, gibt Amazon GameLift Servers die ersten 10 Spielersitzungs-Datensätze zurück, die den Anforderungskriterien entsprechen.

```
// Set request parameters  
Aws::GameLift::Server::Model::DescribePlayerSessionsRequest request;  
request.SetPlayerSessionStatusFilter(Aws::GameLift::Server::Model::PlayerSessionStatusMapper::G  
request.SetLimit(10);  
request.SetGameSessionId("the game session ID");    // can use GetGameSessionId()  
  
// Call DescribePlayerSessions  
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =  
    Aws::GameLift::Server::DescribePlayerSessions(request);
```

GetGameSessionId()

Ruft den eindeutigen Bezeichner der Spielsitzung ab, die derzeit von dem Serverprozess gehostet wird, wenn der Serverprozess aktiv ist. Der Bezeichner wird im ARN-Format zurückgegeben: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`.

Bei Prozessen im Leerlauf, die während einer Spielsitzung noch nicht aktiviert wurden, gibt der Aufruf `Success = True` und `GameSessionId = ""` (eine leere Zeichenfolge) zurück.

Syntax

```
AwsStringOutcome GetGameSessionId();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als `AwsStringOutcome`-Objekt zurück. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetInstanceCertificate()

Ruft den Dateispeicherort eines PEM-codierten TLS-Zertifikats ab, das der Flotte und ihren Instanzen zugeordnet ist. AWS Certificate Manager generiert dieses Zertifikat, wenn Sie eine neue Flotte erstellen, bei der die Zertifikatskonfiguration auf `GENERATED` gesetzt ist. Verwenden Sie dieses Zertifikat, um eine sichere Verbindung mit einem Spiele-Client herzustellen und die Client-/Serverkommunikation zu verschlüsseln.

Syntax

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein `GetInstanceCertificateOutcome` Objekt zurückgegeben, das den Speicherort der TLS-Zertifikatsdatei und der Zertifikatskette der Flotte enthält, die auf der Instanz gespeichert sind. Eine Stammzertifikatsdatei, die aus der Zertifikatskette extrahiert wurde, wird ebenfalls auf der Instanz gespeichert. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Weitere Informationen zu den Daten des Zertifikats und der Zertifikatskette finden Sie unter [GetCertificate Response Elements](#) in der AWS Certificate Manager API-Referenz.

Beispiel

```
Aws::GameLift::GetInstanceCertificateOutcome certificateOutcome =  
    Aws::GameLift::Server::GetInstanceCertificate();
```

GetSdkVersion()

Gibt die aktuelle Versionsnummer des verwendeten SDKs zurück.

Syntax

```
AwsStringOutcome GetSdkVersion();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als `AwsStringOutcome`-Objekt zurück. Die zurückgegebene Zeichenfolge enthält nur die Versionsnummer (z. B. „3.1.5“). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```


GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess führt diese Aktion aus, nachdem er einen `onProcessTerminate()` Rückruf vom Dienst erhalten hat. Amazon GameLift Servers Amazon GameLift Servers kann aus den folgenden `onProcessTerminate()` Gründen aufrufen: (1) wenn der Serverprozess einen schlechten Zustand gemeldet hat oder nicht darauf reagiert hat Amazon GameLift Servers, (2) wenn die Instance während eines Scale-Down-Ereignisses beendet wird oder (3) wenn eine Instance aufgrund einer Spot-Unterbrechung beendet wird.

Wenn der Prozess einen `onProcessTerminate()` Rückruf erhalten hat, entspricht der zurückgegebene Wert der geschätzten Kündigungszeit. Wenn der Prozess keinen `onProcessTerminate()` Rückruf erhalten hat, wird eine Fehlermeldung zurückgegeben. Weitere Informationen über das [Herunterfahren eines Serverprozesses](#).

Syntax

```
AwsLongOutcome GetTerminationTime();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird die Abschlusszeit als `AwsLongOutcome` Objekt zurückgegeben. Der Wert ist die Endzeit, ausgedrückt in verstrichenen Ticks seit 0001 00:00:00. Beispielsweise entspricht der Wert für Datum und Uhrzeit 2020-09-13 12:26:40 -000Z 637355968000000000 Ticks. Wenn keine Kündigungszeit verfügbar ist, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

InitSDK()

Initialisiert das Amazon GameLift Servers-SDK. Diese Methode sollte beim Start aufgerufen werden, bevor andere Amazon GameLift Servers-Initialisierungen durchgeführt werden.

Syntax

```
InitSDKOutcome InitSDK();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein `InitSdkOutcome` Objekt zurückgegeben, das angibt, dass der Serverprozess aufrufbereit ist [ProcessReady\(\)](#).

Beispiel

```
Aws::GameLift::Server::InitSDKOutcome initOutcome =  
    Aws::GameLift::Server::InitSDK();
```

ProcessEnding()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess beendet wird. Diese Methode sollte aufgerufen werden, nachdem alle anderen Aufgaben ausgeführt sind, und insbesondere auch alle aktiven Spielsitzungen beendet sind. Diese Methode sollte mit einem Beendigungscode 0 beendet werden. Andere BeendigungsCodes sorgen für eine Ereignisnachricht zum unsauberen Beenden des Prozesses.

Sobald die Methode mit dem Code 0 beendet wurde, können Sie den Prozess mit einem erfolgreichen Exit-Code beenden. Sie können den Prozess auch mit einem Fehlercode beenden. Wenn Sie den Vorgang mit einem Fehlercode beenden, zeigt das Flottenereignis an, dass der Prozess ungewöhnlich beendet wurde (`SERVER_PROCESS_TERMINATED_UNHEALTHY`).

Syntax

```
GenericOutcome ProcessEnding();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
if (outcome.Success)
    exit(0); // exit with success
// otherwise, exit with error code
exit(errorCode);
```

ProcessReady()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess zum Hosten von Spielsitzungen bereit ist. Rufen Sie diese Methode auf, nachdem Sie erfolgreich Einrichtungsaufgaben aufgerufen [InitSDK\(\)](#) und abgeschlossen haben, die erforderlich sind, bevor der Serverprozess eine Spielsitzung hosten kann. Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Dieser Aufruf ist ein synchron. Wenn Sie einen asynchronen Aufruf durchführen möchten, verwenden Sie [ProcessReadyAsync\(\)](#). Weitere Details finden Sie unter [Initialisieren Sie den Serverprozess](#).

Syntax

```
GenericOutcome ProcessReady(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameters

processParameters

Ein [ProcessParameters](#)-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Die Namen von Callback-Methoden, implementiert in dem Code des Spiel-Servers, den der Amazon GameLift Servers-Service zur Kommunikation mit dem Serverprozess aufruft.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu beliebigen spielsitzungsspezifischen Dateien, die Amazon GameLift Servers erfassen und speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des [ProcessReady\(\)](#)-Aufrufs und der Callback-Funktion.

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");           // Example of a log file written by the
game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);

int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
        listenPort,
        Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcome outcome =
    Aws::GameLift::Server::ProcessReady(processReadyParameter);

// Implement callback functions
void Server::onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome =
        Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void Server::onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
```

```
// such as notifying players, preserving game state data, and other cleanup
GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool Server::onHealthCheck()
{
    bool health;
    // complete health evaluation within 60 seconds and set health
    return health;
}
```

ProcessReadyAsync()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess zum Hosten von Spielsitzungen bereit ist. Diese Methode sollte aufgerufen werden, sobald der Serverprozess zum Hosten einer Spielsitzung bereit ist. Die Parameter geben die Namen von Callback-Funktionen an, die Amazon GameLift Servers unter bestimmten Umständen aufrufen kann. Diese Funktionen müssen im Spiel-Servercode implementiert sein.

Dieser Aufruf ist asynchroner. Wenn Sie einen synchronen Aufruf durchführen möchten, verwenden Sie [ProcessReady\(\)](#). Weitere Details finden Sie unter [Initialisieren Sie den Serverprozess](#).

Syntax

```
GenericOutcomeCallable ProcessReadyAsync(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameters

processParameters

Ein [ProcessParameters](#)-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Die Namen von Callback-Methoden, implementiert in dem Code des Spiel-Servers, den der Amazon GameLift Servers-Service zur Kommunikation mit dem Serverprozess aufruft.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu beliebigen spielsitzungsspezifischen Dateien, die Amazon GameLift Servers erfassen und speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");          // This is an example of a log file
                                                // written by the game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);

int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
        listenPort,
        Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcomeCallable outcome =
    Aws::GameLift::Server::ProcessReadyAsync(processReadyParameter);

// Implement callback functions
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool onHealthCheck()
{
    // perform health evaluation and complete within 60 seconds
```

```
    return health;
}
```

RemovePlayerSession()

Benachrichtigt den Amazon GameLift Servers-Service, dass ein Spieler mit der angegebenen Sitzungs-ID eine Verbindung zu dem Server getrennt hat. Als Reaktion ändert Amazon GameLift Servers den Status des Spielerplatzes in „verfügbar“, so dass er einem neuen Spieler zugewiesen werden kann.

Syntax

```
GenericOutcome RemovePlayerSession(
    const std::string& playerSessionId);
```

Parameters

playerSessionId

Eindeutige ID, die vom Amazon GameLift Servers Dienst als Antwort auf einen Aufruf der AWS Amazon GameLift Servers SDK-API-Aktion ausgegeben wurde [CreatePlayerSession](#). Der Spielclient verweist auf diese ID, wenn er eine Verbindung zum Serverprozess herstellt.

Typ: `std::string`

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
Aws::GameLift::GenericOutcome disconnectOutcome =
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Siehe auch die AWS SDK-Aktion [StartMatchBackfill\(\)](#). Mit dieser Aktion

können Match-Backfill-Anforderungen von einem Game-Server-Prozess initiiert werden, der die Spielsitzung hostet. Erfahren Sie mehr über die [FlexMatchBackfill-Funktion](#).

Diese Aktion ist asynchron. Wenn neue Spieler erfolgreich gematcht werden, liefert der Amazon GameLift Servers Dienst aktualisierte Matchmaker-Daten, indem er die Callback-Funktion aufruft. `OnUpdateGameSession()`

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst [StopMatchBackfill\(\)](#) auf, um die ursprüngliche Anforderung abubrechen.

Syntax

```
StartMatchBackfillOutcome StartMatchBackfill (
    const Aws::GameLift::Server::Model::StartMatchBackfillRequest
    &startBackfillRequest);
```

Parameters

StartMatchBackfillRequest

Ein [StartMatchBackfillRequest](#)-Objekt, das die folgenden Informationen übermittelt:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Informationen sind optional. Bei fehlender ID-Angabe wird automatisch eine ID von Amazon GameLift Servers generiert.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert kann aus den Matchmaker-Daten der Spielsitzung abgerufen werden.
- Die ID der Spielsitzung für das Backfill.
- Verfügbare Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Erforderlich: Ja

Rückgabewert

Gibt ein `StartMatchBackfillOutcome` Objekt mit dem Match-Backfill-Ticket oder einem Fehler mit einer Fehlermeldung zurück. Der Ticketstatus kann mit der AWS SDK-Aktion [DescribeMatchmaking\(\)](#) verfolgt werden.

Beispiel

```
// Build a backfill request
std::vector<Player> players;
Aws::GameLift::Server::Model::StartMatchBackfillRequest startBackfillRequest;
startBackfillRequest.SetTicketId("a ticket ID");
    //optional, autogenerated if not provided
startBackfillRequest.SetMatchmakingConfigurationArn("the matchmaker configuration
    ARN"); //from the game session matchmaker data
startBackfillRequest.SetGameSessionArn("the game session ARN");
    // can use GetGameSessionId()
startBackfillRequest.SetPlayers(players);
    //from the game session matchmaker data

// Send backfill request
Aws::GameLift::StartMatchBackfillOutcome backfillOutcome =
    Aws::GameLift::Server::StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void Server::OnUpdateGameSession(Aws::GameLift::Server::Model::GameSession gameSession,
    Aws::GameLift::Server::Model::UpdateReason updateReason, std::string backfillTicketId)
{
    // handle status messages
    // perform game-specific tasks to prep for newly matched players
}
```

StopMatchBackfill()

Bricht eine aktive mit dem Befehl [StartMatchBackfill\(\)](#) erstellte Match-Backfill-Anforderung ab. Siehe auch die AWS SDK-Aktion [StopMatchmaking\(\)](#). Erfahren Sie mehr über die [FlexMatchBackfill-Funktion](#).

Syntax

```
GenericOutcome StopMatchBackfill (
    const Aws::GameLift::Server::Model::StopMatchBackfillRequest &stopBackfillRequest);
```

Parameters

StopMatchBackfillRequest

Ein [StopMatchBackfillRequest](#)-Objekt, das das abzubrechende Matchmaking-Ticket identifiziert:

- Ticket-ID der abzubrechenden Backfill-Anforderung
- Matchmaker, an den die Backfill-Anforderung gesendet wurde
- Spielsitzung für die Backfill-Anforderung

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
// Set backfill stop request parameters

Aws::GameLift::Server::Model::StopMatchBackfillRequest stopBackfillRequest;
stopBackfillRequest.SetTicketId("the ticket ID");
stopBackfillRequest.SetGameSessionArn("the game session ARN");
    // can use GetGameSessionId()
stopBackfillRequest.SetMatchmakingConfigurationArn("the matchmaker configuration ARN");
    // from the game session matchmaker data

Aws::GameLift::GenericOutcome stopBackfillOutcome =
    Aws::GameLift::Server::StopMatchBackfillRequest(stopBackfillRequest);
```

TerminateGameSession()

Diese Methode ist mit Version 4.0.1 veraltet. Stattdessen sollte der Serverprozess [ProcessEnding\(\)](#) nach dem Ende einer Spielsitzung aufgerufen werden.

Informiert den Amazon GameLift Servers Dienst darüber, dass der Serverprozess die aktuelle Spielsitzung beendet hat. Diese Aktion wird aufgerufen, wenn der Serverprozess aktiv bleibt und bereit ist, eine neue Spielsitzung zu veranstalten. Sie sollte erst aufgerufen werden, nachdem der Vorgang zum Beenden der Spielsitzung abgeschlossen ist, da sie signalisiert Amazon GameLift Servers, dass der Serverprozess sofort für die Durchführung einer neuen Spielsitzung verfügbar ist.

Diese Aktion wird nicht aufgerufen, wenn der Serverprozess nach dem Ende der Spielsitzung heruntergefahren wird. Rufen Sie stattdessen auf, [ProcessEnding\(\)](#) um zu signalisieren, dass sowohl die Spielsitzung als auch der Serverprozess beendet werden.

Syntax

```
GenericOutcome TerminateGameSession();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spiellersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt. Siehe auch die AWS SDK-Aktion [UpdateGameSession\(\)](#).

Syntax

```
GenericOutcome UpdatePlayerSessionCreationPolicy(  
    Aws::GameLift::Model::PlayerSessionCreationPolicy newPlayerSessionPolicy);
```

Parameters

newPlayerSessionRichtlinie

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Typ: `Aws::GameLift::Model::PlayerSessionCreationPolicy` enum. Gültige Werte sind:

- `ACCEPT_ALL` – Akzeptiert alle neuen Spiellersitzungen.
- `DENY_ALL` – Verwehrt neue Spiellersitzungen.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
Aws::GameLift::GenericOutcome outcome =  
    Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

Zerstören ()

Löscht den Speicher, der von `initSDK ()` während der Initialisierung des Spieleservers zugewiesen wurde. Verwenden Sie diese Methode, nachdem Sie einen Spieleserver-Prozess beendet haben, um zu vermeiden, dass Serverspeicher verschwendet wird.

Syntax

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

Parameters

Es gibt keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird der von `initSDK` zugewiesene Speicher bereinigt, nachdem ein Spieleserverprozess beendet wurde.

```
if (Aws::GameLift::Server::ProcessEnding().IsSuccess()) {  
    Aws::GameLift::Server::Destroy();  
    exit(0);  
}
```

C#-Server-SDK für Amazon GameLift Servers 4.x — Aktionen

Verwenden Sie die Server-SDK-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrieren. Amazon GameLift Servers Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

 Note

Diese Referenz bezieht sich auf eine frühere Version des Server-SDK für Amazon GameLift Servers. Informationen zur neuesten Version finden Sie unter [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#).

C#-Server-SDK für Amazon GameLift Servers 4.x -- Datentypen

Verwenden Sie die Server-SDK-Referenz, um Ihr Multiplayer-Spiel zum Hosten zu integrieren Amazon GameLift Servers. Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

 Note

Diese Referenz bezieht sich auf eine frühere Version des Server-SDK für Amazon GameLift Servers. Informationen zur neuesten Version finden Sie unter [C#-Server-SDK 5.x für Amazon GameLift Servers -- Datentypen](#).

[C#-Server-SDK für Amazon GameLift Servers 4.x — Aktionen](#)

Themen

- [LogParameters](#)
- [DescribePlayerSessionsRequest](#)
- [ProcessParameters](#)
- [StopMatchBackfillRequest](#)

LogParameters

Dieser Datentyp wird verwendet, um zu identifizieren, welche Dateien während einer Sitzung generiert werden, die Sie in Amazon GameLift Servers hochladen und nach Beendigung der Spielsitzung speichern möchten. Diese Informationen werden dem Amazon GameLift Servers-Service in einem [ProcessReady\(\)](#)-Aufruf übermittelt.

Inhalt

logPaths

Liste der Verzeichnispfade zu Spiel-Server-Protokolldateien, die Amazon GameLift Servers für den späteren Zugriff speichern soll. Diese Dateien werden von einem Serverprozess während jeder Spielsitzung generiert; Dateipfade und -namen werden in Ihrem Spiel-Server definiert und im Spiel-Build-Stammverzeichnis gespeichert. Die Protokollpfade müssen absolut sein. Zum Beispiel: Wenn Ihr Spiel-Build Spielsitzungsprotokolle in einem Pfad wie `MyGame\sessionlogs` speichert, ist der Protokollpfad `c:\game\MyGame\sessionLogs` (auf einer Windows-Instance) oder `/local/game/MyGame/sessionLogs` (auf einer Linux-Instance).

Typ: `Liste<String>`

Erforderlich: Nein

DescribePlayerSessionsRequest

Dieser Datentyp wird verwendet, um anzugeben, welche Spielsitzung(en) abgerufen werden sollen. Es kann auf verschiedene Arten verwendet werden: (1) bietet a, `PlayerSessionId` um eine bestimmte Spielsitzung anzufordern; (2) bietet a, `GameSessionId` um alle Spielsitzungen in der angegebenen Spielsitzung anzufordern; oder (3) bietet a, `PlayerId` um alle Spielsitzungen für den angegebenen Spieler anzufordern. Verwenden Sie für große Sammlungen von Spielsitzungen die Paginierungsparameter, um Ergebnisse als aufeinander folgende Seiten abzurufen.

Inhalt

GameSessionId

Eindeutiger Bezeichner für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielsitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist wie folgt: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. Der Wert von `<ID string>` ist entweder eine benutzerdefinierte ID-Zeichenfolge (sofern bei der Erstellung der Spielsitzung angegeben) oder eine generierte Zeichenfolge.

Typ: Zeichenfolge

Erforderlich: Nein

Limit

Maximale Anzahl der zurückzugebenden Ergebnisse. Verwenden Sie diesen Parameter mit `NextToken`, um Ergebnisse in Form einer Reihe von aufeinanderfolgenden Seiten zu erhalten. Wenn eine Spielsitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Ganzzahl

Erforderlich: Nein

NextToken

Token, der den Beginn der nächsten Seite mit Ergebnissen anzeigt. Verwenden Sie den Token, der bei einem früheren Aufruf dieser Aktion zurückgegeben wurde. Um den Beginn des Ergebnissatzes anzugeben, geben Sie keinen Wert an. Wenn eine Spielsitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Zeichenfolge

Erforderlich: Nein

PlayerId

Eindeutiger Bezeichner für den Spieler. Spieler IDs werden vom Entwickler definiert. Siehe [Spieler generieren IDs](#).

Typ: Zeichenfolge

Erforderlich: Nein

PlayerSessionId

Eindeutiger Bezeichner für eine Spielsitzung.

Typ: Zeichenfolge

Erforderlich: Nein

PlayerSessionStatusFilter

Status der Spielsitzung für die Filterung der Ergebnisse. Mögliche Spielsitzungsstatus sind u. a.:

- **RESERVIERT** — Die Anfrage für die Spielsitzung wurde empfangen, aber der Spieler hat sich noch nicht mit dem Serverprozess verbunden, der bestätigt and/or wurde.
- **ACTIVE** – Der Spieler wurden vom Serverprozess validiert und ist derzeit verbunden.

- COMPLETED – Die Spielerverbindung wurde beendet.
- TIMEDOUT — Es wurde eine Anfrage für eine Spielsitzung empfangen, aber der Spieler hat keine Verbindung hergestellt und das Timeout-Limit (60 Sekunden) and/or wurde nicht bestätigt.

Typ: Zeichenfolge

Erforderlich: Nein

ProcessParameters

Dieser Datentyp enthält den Parametersatz, der in einem [ProcessReady\(\)](#)-Aufruf an den Amazon GameLift Servers-Service gesendet wurde.

Inhalt

port

Nummer des Ports, den der Serverprozess für neue Spielerverbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: Ganzzahl

Erforderlich: Ja

logParameters

Objekt mit einer Liste der Verzeichnispfade zu den Spielsitzungsprotokolldateien.

Typ: `Aws::GameLift::Server::`[LogParameters](#)

Erforderlich: Ja

onStartGameSitzung

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um eine neue Spielsitzung zu aktivieren. Amazon GameLift Servers ruft diese Funktion als Antwort auf die Client-Anfrage auf. [CreateGameSession](#) Die Callback-Funktion verwendet ein [GameSession](#) Objekt (definiert in der Amazon GameLift ServersService API-Referenz).

Typ: `void OnStartGameSessionDelegate(GameSession gameSession)`

Erforderlich: Ja

onProcessTerminate

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion wartet Amazon GameLift Servers fünf Minuten auf die Beendigung des Serverprozesses und antwortet mit einem [ProcessEnding\(\)](#)-Aufruf, bevor es den Serverprozess beendet.

Typ: void OnProcessTerminateDelegate()

Erforderlich: Ja

onHealthCheck

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift Servers ruft diese Funktion alle 60 Sekunden auf. Nach dem Aufruf dieser Funktion wartet Amazon GameLift Servers 60 Sekunden auf eine Antwort; wenn keine Antwort eingeht, wird der Serverprozess als fehlerhaft notiert.

Typ: bool OnHealthCheckDelegate()

Erforderlich: Ja

onUpdateGameSitzung

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift Servers ruft diese Funktion auf, wenn eine [Match-Backfill-Anfrage](#) bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Sie übergibt ein [GameSession](#)-Objekt, ein Status-Update (updateReason) und die Match-Backfill-Ticket-ID.

Typ: void OnUpdateGameSessionDelegate (UpdateGameSession
updateGameSession)

Erforderlich: Nein

StartMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung zu versenden. Die Informationen werden dem Amazon GameLift Servers-Service in einem [StartMatchBackfill\(\)](#)-Aufruf übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner für die Spielsitzung. Die SDK-Methode [GetGameSessionId\(\)](#) gibt den Bezeichner im ARN-Format zurück.

Typ: Zeichenfolge

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner in Form eines ARN für den Matchmaker, der für diese Anfrage verwendet werden soll. Um den Matchmaker zu finden, der für die Erstellung der ursprünglichen Spielsitzung verwendet wurde, suchen Sie im Spielsitzungsobjekt in der Dateneigenschaft Matchmaker. Weitere Informationen zu Matchmaker-Daten finden Sie unter [Arbeiten mit Matchmaker-Daten](#).

Typ: Zeichenfolge

Erforderlich: Ja

Players

Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Eine Beschreibung des Player-Objektformats finden Sie im Amazon GameLift ServersAPI-Referenzhandbuch. Spielerattribute und Teamzuweisungen finden Sie im Objekt „Spielsitzung“ in der Eigenschaft „Matchmaker-Daten“. IDs Wenn der Matchmaker die Latenz nutzt, erfassen Sie die aktualisierte Latenz für die aktuelle Region und nehmen Sie sie in die Daten jedes Spielers auf.

Typ: [Player](#)[]

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner für ein Matchmaking- oder Match-Backfill-Request-Ticket. Wenn hier kein Wert angegeben wird, Amazon GameLift Servers wird einer in Form einer UUID generiert. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abubrechen.

Typ: Zeichenfolge

Erforderlich: Nein

StopMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung abzuberechnen. Die Informationen werden dem Amazon GameLift Servers-Service in einem [StopMatchBackfill\(\)](#)-Aufruf übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner der Spielsitzung für die Abbruchanfrage.

Typ: Zeichenfolge

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner des Matchmakers, an den diese Anfrage gesendet wurde.

Typ: Zeichenfolge

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner des abzuberechnenden Tickets für die Backfill-Anforderung.

Typ: Zeichenfolge

Erforderlich: Ja

C#-Server-SDK für Amazon GameLift Servers 4.x -- Datentypen

Themen

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)

- [GetSdkVersion\(\)](#)
- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

Informiert den Amazon GameLift Servers Dienst darüber, dass ein Spieler mit der angegebenen Spielsitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Bestätigung benötigt. Amazon GameLift Servers überprüft, ob die Spielsitzungs-ID gültig ist, d. h. ob die Spieler-ID einen Spielerplatz in der Spielsitzung reserviert hat. Nach der Bestätigung ändert Amazon GameLift Servers den Status des Spielerplatzes von RESERVED nach ACTIVE.

Syntax

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Parameters

playerSessionId

Eindeutige ID, die ausgestellt wird Amazon GameLift Servers, wenn eine neue Spielsitzung erstellt wird. In einem `PlayerSession` Objekt ist eine Spielsitzungs-ID angegeben, die als Antwort auf einen Client-Aufruf der GameLift API-Aktionen [StartGameSessionPlacement](#), [CreateGameSession](#) [DescribeGameSessionPlacement](#), oder zurückgegeben wird [DescribePlayerSessions](#).

Typ: Zeichenfolge

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel zeigt eine Funktion zur Bearbeitung einer Verbindungsanfrage, einschließlich der Validierung und Ablehnung einer ungültigen Spielsitzung. IDs

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerId){
    var acceptPlayerSessionOutcome =
    GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
    if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerId);
        connection.Accept();
    }
    else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);    }
}
```

ActivateGameSession()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess eine Sitzungs-ID für das Spiel ist aktiviert hat und nun Verbindungen annehmen kann. Diese Aktion muss im Rahmen der `onStartGameSession()`-Callback-Funktion aufgerufen werden, nachdem die Initialisierung der Spielsitzung vollständig abgeschlossen ist.

Syntax

```
GenericOutcome ActivateGameSession()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel wird `ActivateGameSession()` als Teil der `onStartGameSession()`-Delegate-Funktion aufgerufen.

```
void OnStartGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map

    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

DescribePlayerSessions()

Ruft Sitzungsdaten der Spieler, einschließlich Einstellungen, Metadaten zu der Sitzung und Spielerdaten ab. Verwenden Sie diese Aktion, um für eine einzelne Spilersitzung, für alle Spilersitzungen in einer Spielsitzung oder für alle Spilersitzungen zu einer einzelnen Spieler-ID abzurufen.

Syntax

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
describePlayerSessionsRequest)
```

Parameters

describePlayerSessionsAnfrage

Ein [DescribePlayerSessionsRequest](#)-Objekt, mit dem beschrieben wird, welche Spilersitzungen abgerufen werden sollen.

Erforderlich: Ja

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein `DescribePlayerSessionsOutcome`-Objekt mit einer Menge von Spilersitzungsobjekten zurück, die den Anforderungsparametern entsprechen. Player-Sitzungsobjekte haben eine Struktur, die mit dem AWS Amazon GameLift Servers [PlayerSession](#) SDK-API-Datentyp identisch ist.

Beispiel

Dieses Beispiel zeigt eine Anforderung für alle Spilersitzungen, die derzeit aktiv mit einer angegebenen Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen `NextToken` und ihn auf 10 setzen, Amazon GameLift Servers werden die Datensätze der ersten 10 Spilersitzungen zurückgegeben, die der Anfrage entsprechen.

```
// Set request parameters
var describePlayerSessionsRequest = new
    Aws.GameLift.Server.Model.DescribePlayerSessionsRequest()
{
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    //gets the ID for
    the current game session
    Limit = 10,
    PlayerSessionStatusFilter =
    PlayerSessionStatusMapper.GetNameForPlayerSessionStatus(PlayerSessionStatus.ACTIVE)
};
// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::Model::DescribePlayerSessions(describePlayerSessionRequest);
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die derzeit von dem Serverprozess gehostet wird, wenn der Serverprozess aktiv ist.

Bei Prozessen im Leerlauf, die während einer Spielsitzung noch nicht aktiviert wurden, gibt der Aufruf `Success = True` und `GameSessionId = ""` (eine leere Zeichenfolge) zurück.

Syntax

```
AwsStringOutcome GetGameSessionId()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als `AwsStringOutcome`-Objekt zurück. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
var getSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetInstanceCertificate()

Ruft den Dateispeicherort eines PEM-codierten TLS-Zertifikats ab, das der Flotte und ihren Instanzen zugeordnet ist. AWS Certificate Manager generiert dieses Zertifikat, wenn Sie eine neue Flotte erstellen, bei der die Zertifikatskonfiguration auf GENERATED gesetzt ist. Verwenden Sie dieses Zertifikat, um eine sichere Verbindung mit einem Spiele-Client herzustellen und die Client-/Serverkommunikation zu verschlüsseln.

Syntax

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein `GetInstanceCertificateOutcome` Objekt zurückgegeben, das den Speicherort der TLS-Zertifikatsdatei und der Zertifikatskette der Flotte enthält, die auf der Instanz gespeichert sind. Eine Stammzertifikatsdatei, die aus der Zertifikatskette extrahiert wurde, wird ebenfalls auf der Instanz gespeichert. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Weitere Informationen zu den Daten des Zertifikats und der Zertifikatskette finden Sie unter [GetCertificate Response Elements](#) in der AWS Certificate Manager API-Referenz.

Beispiel

```
var getInstanceCertificateOutcome = GameLiftServerAPI.GetInstanceCertificate();
```

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
AwsStringOutcome GetSdkVersion()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als `AwsStringOutcome`-Objekt zurück. Die zurückgegebene Zeichenfolge enthält nur die Versionsnummer (z. B. „3.1.5“). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess führt diese Aktion aus, nachdem er einen `onProcessTerminate()` Rückruf vom Dienst erhalten hat. Amazon GameLift Servers Amazon GameLift Servers kann aus den folgenden `onProcessTerminate()` Gründen aufrufen: (1) bei schlechtem Zustand (der Serverprozess hat den Zustand des Ports gemeldet oder nicht darauf reagiert) Amazon GameLift Servers, (2) wenn die Instance während eines Scale-Down-Ereignisses beendet wird oder (3) wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.

Wenn der Prozess einen `onProcessTerminate()` Rückruf erhalten hat, entspricht der zurückgegebene Wert der geschätzten Endzeit. Wenn der Prozess keinen `onProcessTerminate()` Rückruf erhalten hat, wird eine Fehlermeldung zurückgegeben. Weitere Informationen über das [Herunterfahren eines Serverprozesses](#).

Syntax

```
AwsDateTimeOutcome GetTerminationTime()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird die Abschlusszeit als `AwsDateTimeOutcome` Objekt zurückgegeben. Der Wert ist die Endzeit, ausgedrückt in verstrichenen Ticks seit 0001 00:00:00. Beispielsweise entspricht der Wert für Datum und Uhrzeit 2020-09-13 12:26:40 -000Z 637355968000000000 Ticks. Wenn keine Kündigungszeit verfügbar ist, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
var getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

InitSDK()

Initialisiert das Amazon GameLift Servers-SDK. Diese Methode sollte beim Start aufgerufen werden, bevor andere Amazon GameLift Servers-Initialisierungen durchgeführt werden.

Syntax

```
InitSDKOutcome InitSDK()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein `InitSdkOutcome` Objekt zurückgegeben, das angibt, dass der Serverprozess aufrufbereit ist [ProcessReady\(\)](#).

Beispiel

```
var initSDKOutcome = GameLiftServerAPI.InitSDK();
```

ProcessEnding()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess beendet wird. Diese Methode sollte aufgerufen werden, nachdem alle anderen Aufgaben ausgeführt sind, und insbesondere auch alle aktiven Spielsitzungen beendet sind. Diese Methode sollte mit einem

Beendigungscode 0 beendet werden. Andere Beendigungscode sorgen für eine Ereignisnachricht zum unsauberen Beenden des Prozesses.

Sobald die Methode mit dem Code 0 beendet wurde, können Sie den Prozess mit einem erfolgreichen Exit-Code beenden. Sie können den Prozess auch mit einem Fehlercode beenden. Wenn Sie den Vorgang mit einem Fehlercode beenden, zeigt das Flottenereignis an, dass der Prozess ungewöhnlich beendet wurde (SERVER_PROCESS_TERMINATED_UNHEALTHY).

Syntax

```
GenericOutcome ProcessEnding()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
    Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

ProcessReady()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess zum Hosten von Spielsitzungen bereit ist. Rufen Sie diese Methode auf, nachdem Sie erfolgreich Einrichtungsaufgaben aufgerufen [InitSDK\(\)](#) und abgeschlossen haben, die erforderlich sind, bevor der Serverprozess eine Spielsitzung hosten kann. Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

Parameters

processParameters

Ein [ProcessParameters](#)-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Die Namen von Callback-Methoden, implementiert in dem Code des Spiel-Servers, den der Amazon GameLift Servers-Service zur Kommunikation mit dem Serverprozess aufruft.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu beliebigen spielsitzungsspezifischen Dateien, die Amazon GameLift Servers erfassen und speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des [ProcessReady\(\)](#)-Aufrufs und der Delegate-Funktion.

```
// Set parameters and call ProcessReady
var processParams = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    new LogParameters(new List<string>()           // Examples of log and error files
        written by the game server
        {
            "C:\\game\\logs",
            "C:\\game\\error"
        })
);

var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);

// Implement callback functions
```

```
void OnGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}

void OnProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    var ProcessEndingOutcome = GameLiftServerAPI.ProcessEnding();
}

bool OnHealthCheck()
{
    bool isHealthy;
    // complete health evaluation within 60 seconds and set health
    return isHealthy;
}
```

RemovePlayerSession()

Benachrichtigt den Amazon GameLift Servers-Service, dass ein Spieler mit der angegebenen Sitzungs-ID eine Verbindung zu dem Server getrennt hat. Als Reaktion ändert Amazon GameLift Servers den Status des Spielerplatzes in „verfügbar“, so dass er einem neuen Spieler zugewiesen werden kann.

Syntax

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Parameters

playerSessionId

Eindeutige IDAmazon GameLift Servers, die ausgegeben wird, wenn eine neue Spilersitzung erstellt wird. In einem `PlayerSession` Objekt ist eine Spilersitzungs-ID angegeben, die als Antwort auf einen Client-Aufruf der GameLift API-Aktionen [StartGameSessionPlacement](#), [CreateGameSession](#) [DescribeGameSessionPlacement](#), oder zurückgegeben wird [DescribePlayerSessions](#).

Typ: Zeichenfolge

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Siehe auch die AWS SDK-Aktion [StartMatchBackfill\(\)](#). Mit dieser Aktion können Match-Backfill-Anforderungen von einem Game-Server-Prozess initiiert werden, der die Spielsitzung hostet. Erfahren Sie mehr über die [FlexMatchBackfill-Funktion](#).

Diese Aktion ist asynchron. Wenn neue Spieler erfolgreich gematcht werden, liefert der Amazon GameLift Servers-Service über die Callback-Funktion `OnUpdateGameSession()` aktualisierte Matchmaker-Daten.

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst [StopMatchBackfill\(\)](#) auf, um die ursprüngliche Anforderung abubrechen.

Syntax

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest  
    startBackfillRequest);
```

Parameters

StartMatchBackfillRequest

Ein [StartMatchBackfillRequest](#)-Objekt, das die folgenden Informationen übermittelt:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Informationen sind optional. Bei fehlender ID-Angabe wird automatisch eine ID von Amazon GameLift Servers generiert.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert kann aus den Matchmaker-Daten der Spielsitzung abgerufen werden.
- Die ID der Spielsitzung für das Backfill.
- Verfügbare Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Erforderlich: Ja

Rückgabewert

Gibt ein `StartMatchBackfillOutcome` Objekt mit der passenden Backfill-Ticket-ID zurück oder schlägt fehl, und es wird eine Fehlermeldung angezeigt.

Beispiel

```
// Build a backfill request
var startBackfillRequest = new AWS.GameLift.Server.Model.StartMatchBackfillRequest()
{
    TicketId = "a ticket ID", //optional
    MatchmakingConfigurationArn = "the matchmaker configuration ARN",
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result, // gets ID for
current game session
    //get player data for all currently connected players
    MatchmakerData matchmakerData =
        MatchmakerData.FromJson(gameSession.MatchmakerData); // gets matchmaker
data for current players
    // get matchmakerData.Players
    // remove data for players who are no longer connected
    Players = ListOfPlayersRemainingInTheGame
};

// Send backfill request
var startBackfillOutcome = GameLiftServerAPI.StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void OnUpdateGameSession(GameSession myGameSession)
{
    // game-specific tasks to prepare for the newly matched players and update
    matchmaker data as needed
}
```

```
}
```

StopMatchBackfill()

Bricht eine aktive mit dem Befehl [StartMatchBackfill\(\)](#) erstellte Match-Backfill-Anforderung ab. Siehe auch die AWS SDK-Aktion [StopMatchmaking\(\)](#). Erfahren Sie mehr über die [FlexMatchBackfill-Funktion](#).

Syntax

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameters

StopMatchBackfillRequest

Ein [StopMatchBackfillRequest](#)-Objekt, das das abzubrechende Matchmaking-Ticket identifiziert:

- Ticket-ID der abzubrechenden Backfill-Anforderung
- Matchmaker, an den die Backfill-Anforderung gesendet wurde
- Spielsitzung für die Backfill-Anforderung

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

```
// Set backfill stop request parameters

var stopBackfillRequest = new AWS.GameLift.Server.Model.StopMatchBackfillRequest()
{
    TicketId = "a ticket ID", //optional, if not provided one is autogenerated
    MatchmakingConfigurationArn = "the matchmaker configuration ARN", //from the game
    session matchmaker data
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result //gets the ID for
    the current game session
}
```



```
};  
  
var stopBackfillOutcome =  
    GameLiftServerAPI.StopMatchBackfillRequest(stopBackfillRequest);
```

TerminateGameSession()

Diese Methode ist mit Version 4.0.1 veraltet. Stattdessen sollte der Serverprozess [ProcessEnding\(\)](#) nach dem Ende einer Spielsitzung aufgerufen werden.

Informiert den Amazon GameLift Servers Dienst darüber, dass der Serverprozess die aktuelle Spielsitzung beendet hat. Diese Aktion wird aufgerufen, wenn der Serverprozess aktiv bleibt und bereit ist, eine neue Spielsitzung zu veranstalten. Sie sollte erst aufgerufen werden, nachdem der Vorgang zum Beenden der Spielsitzung abgeschlossen ist, da sie signalisiert Amazon GameLift Servers, dass der Serverprozess sofort für die Durchführung einer neuen Spielsitzung verfügbar ist.

Diese Aktion wird nicht aufgerufen, wenn der Serverprozess nach dem Ende der Spielsitzung heruntergefahren wird. Rufen Sie stattdessen auf, [ProcessEnding\(\)](#) um zu signalisieren, dass sowohl die Spielsitzung als auch der Serverprozess beendet werden.

Syntax

```
GenericOutcome TerminateGameSession()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

Dieses Beispiel zeigt einen Serverprozess am Ende einer Spielsitzung.

```
// game-specific tasks required to gracefully shut down a game session,  
// such as notifying players, preserving game state data, and other cleanup  
  
var terminateGameSessionOutcome = GameLiftServerAPI.TerminateGameSession();
```

```
var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spilersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt. (Siehe auch die Aktion [UpdateGameSession\(\)](#) in der Amazon GameLift ServersService-API-Referenz).

Syntax

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy  
playerSessionPolicy)
```

Parameters

newPlayerSessionRichtlinie

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Eingabe: [PlayerSessionCreationPolicy](#) enum. Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spilersitzungen.
- DENY_ALL – Verwehrt neue Spilersitzungen.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
var updatePlayerSessionCreationPolicyOutcomex =  
  
GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

Server-SDK (Unreal) für Amazon GameLift Servers -- Aktionen

Verwenden Sie das Server-SDK für Unreal, um Ihr Multiplayer-Spiel zum Hosten zu integrieren. Amazon GameLift Servers Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Note

Diese Referenz bezieht sich auf eine frühere Version des Server-SDK für Amazon GameLift Servers. Informationen zur neuesten Version finden Sie unter [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#).

Diese API ist in `GameLiftServerSDK.h` und `GameLiftServerSDKModels.h` definiert.

So richten Sie das Unreal Engine-Plug-in ein und zeigen Codebeispiele an [Amazon GameLift ServersIn ein Unreal Engine-Projekt integrieren](#):

Server-SDK (Unreal) für Amazon GameLift Servers -- Datentypen

Verwenden Sie die Referenz zum Amazon GameLift Servers Server-SDK für Unreal, um Ihr Multiplayer-Spiel zum Hosten zu integrieren. Amazon GameLift Servers Hinweise zum Integrationsprozess finden Sie unter [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#).

Note

Diese Referenz bezieht sich auf eine frühere Version des Server-SDK für Amazon GameLift Servers. Informationen zur neuesten Version finden Sie unter [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Datentypen](#).

Diese API ist in `GameLiftServerSDK.h` und `GameLiftServerSDKModels.h` definiert.

So richten Sie das Unreal Engine-Plug-in ein und zeigen Codebeispiele an [Amazon GameLift ServersIn ein Unreal Engine-Projekt integrieren](#):

[Server-SDK \(Unreal\) für Amazon GameLift Servers -- Aktionen](#)

Themen

- [FDescribePlayerSessionsRequest](#)
- [FProcessParameter](#)
- [FStartMatchBackfillRequest](#)
- [FStopMatchBackfillRequest](#)

FDescribePlayerSessionsRequest

Dieser Datentyp wird verwendet, um anzugeben, welche Spielsitzung(en) abgerufen werden sollen. Sie können ihn wie folgt verwenden:

- Geben Sie a `PlayerSessionId` an, um eine bestimmte Spielsitzung anzufordern.
- Geben Sie a `GameSessionId` an, um alle Spielsitzungen in der angegebenen Spielsitzung anzufordern.
- Geben Sie a `PlayerId` an, um alle Spielsitzungen für den angegebenen Spieler anzufordern.

Verwenden Sie für große Sammlungen von Spielsitzungen die Paginierungsparameter, um Ergebnisse in aufeinander folgenden Blöcken abzurufen.

Inhalt

GameSessionId

Eindeutiger Bezeichner für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielsitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist wie folgt: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. Der Wert von `<ID-String>` ist entweder eine benutzerdefinierte ID-Zeichenfolge (sofern bei der Erstellung der Spielsitzung angegeben) oder eine generierte Zeichenfolge.

Typ: Zeichenfolge

Erforderlich: Nein

Limit

Maximale Anzahl der zurückzugebenden Ergebnisse. Verwenden Sie diesen Parameter mit `NextToken`, um Ergebnisse in Form einer Reihe von aufeinanderfolgenden Seiten zu erhalten. Wenn eine Spielsitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Ganzzahl

Erforderlich: Nein

NextToken

Token, der den Beginn der nächsten Seite mit Ergebnissen anzeigt. Verwenden Sie den Token, der bei einem früheren Aufruf dieser Aktion zurückgegeben wurde. Um den Beginn des Ergebnissatzes anzugeben, geben Sie keinen Wert an. Wenn eine Spielsitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Zeichenfolge

Erforderlich: Nein

PlayerId

Eindeutiger Bezeichner für den Spieler. Spieler IDs werden vom Entwickler definiert. Siehe [Spieler generieren IDs](#).

Typ: Zeichenfolge

Erforderlich: Nein

PlayerSessionId

Eindeutiger Bezeichner für eine Spielsitzung.

Typ: Zeichenfolge

Erforderlich: Nein

PlayerSessionStatusFilter

Status der Spielsitzung für die Filterung der Ergebnisse. Mögliche Spielsitzungsstatus sind u. a.:

- RESERVIERT — Die Anfrage für die Spielsitzung wurde empfangen, aber der Spieler hat sich noch nicht mit dem Serverprozess verbunden, der bestätigt and/or wurde.
- ACTIVE – Der Spieler wurden vom Serverprozess validiert und ist derzeit verbunden.
- COMPLETED – Die Spielerverbindung wurde beendet.
- TIMEDOUT — Eine Anfrage für eine Spielsitzung wurde empfangen, aber der Spieler hat keine Verbindung hergestellt und das Timeout-Limit (60 Sekunden) and/or wurde nicht bestätigt.

Typ: Zeichenfolge

Erforderlich: Nein

FProcessParameter

Dieser Datentyp enthält den Parametersatz, der in einem [ProcessReady\(\)](#)-Aufruf an den Amazon GameLift Servers-Service gesendet wurde.

Inhalt

port

Nummer des Ports, den der Serverprozess für neue Spielerverbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: Ganzzahl

Erforderlich: Ja

logParameters

Objekt mit einer Liste der Verzeichnispfade zu den Spielsitzungsprotokolldateien.

Typ: TArray < FString >

Erforderlich: Nein

onStartGameSitzung

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um eine neue Spielsitzung zu aktivieren. Amazon GameLift Servers ruft diese Funktion als Antwort auf die Client-Anfrage auf. [CreateGameSession](#) Die Callback-Funktion verwendet ein [GameSession](#) Objekt (definiert in der Amazon GameLift ServersService API-Referenz).

Typ: FOnStartGameSession

Erforderlich: Ja

onProcessTerminate

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion wartet Amazon GameLift Servers fünf Minuten auf die Beendigung des Serverprozesses und antwortet mit einem [ProcessEnding\(\)](#)-Aufruf, bevor es den Serverprozess beendet.

Typ: Delegierter FSimple

Erforderlich: Nein

onHealthCheck

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift Servers ruft diese Funktion alle 60 Sekunden auf. Nach dem Aufruf dieser Funktion wartet Amazon GameLift Servers 60 Sekunden auf eine Antwort; wenn keine Antwort eingeht, wird der Serverprozess als fehlerhaft notiert.

Typ: FOnHealthCheck

Erforderlich: Nein

onUpdateGameSitzung

Name der Callback-Funktion, die der Amazon GameLift Servers Dienst aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift Servers ruft diese Funktion auf, wenn eine [Match-Backfill-Anfrage](#) bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Sie übergibt ein [GameSession](#)-Objekt, ein Status-Update (updateReason) und die Match-Backfill-Ticket-ID.

Typ: FOnUpdateGameSession

Erforderlich: Nein

FStartMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung zu versenden. Die Informationen werden dem Amazon GameLift Servers-Service in einem [StartMatchBackfill\(\)](#)-Aufruf übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner für die Spielsitzung. Die API-Aktion [GetGameSessionId\(\)](#) gibt den Bezeichner im ARN-Format zurück.

Typ: FString

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner in Form eines ARN für den Matchmaker, der für diese Anfrage verwendet werden soll. Um den Matchmaker zu finden, der für die Erstellung der ursprünglichen Spielsitzung verwendet wurde, suchen Sie im Spielsitzungsobjekt in der Dateneigenschaft Matchmaker.

Weitere Informationen zu Matchmaker-Daten finden Sie unter [Arbeiten mit Matchmaker-Daten](#).

Typ: FString

Erforderlich: Ja

Players

Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Eine Beschreibung des Player-Objektformats finden Sie im Amazon GameLift ServersAPI-Referenzhandbuch. Spielerattribute und Teamzuweisungen finden Sie im Objekt „Spielsitzung“ in der Eigenschaft „Matchmaker-Daten“. IDs Wenn der Matchmaker die Latenz nutzt, erfassen Sie die aktualisierte Latenz für die aktuelle Region und nehmen Sie sie in die Daten jedes Spielers auf.

Geben Sie ein: TArray [< > FPlayer](#)

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner für ein Matchmaking- oder Match-Backfill-Request-Ticket. Wenn hier kein Wert angegeben wird, Amazon GameLift Servers wird einer in Form einer UUID generiert. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abubrechen.

Typ: FString

Erforderlich: Nein

FStopMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung abubrechen. Die Informationen werden dem Amazon GameLift Servers-Service in einem [StopMatchBackfill\(\)](#)-Aufruf übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner der Spielsitzung für die Abbruchanfrage.

Typ: FString

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner des Matchmakers, an den diese Anfrage gesendet wurde.

Typ: FString

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner des abzubrechenden Tickets für die Backfill-Anforderung.

Typ: FString

Erforderlich: Ja

Server-SDK (Unreal) für Amazon GameLift Servers -- Datentypen

Themen

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)

- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

Informiert den Amazon GameLift Servers Dienst darüber, dass ein Spieler mit der angegebenen Spielsitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Bestätigung benötigt. Amazon GameLift Servers überprüft, ob die Spielsitzungs-ID gültig ist, d. h. ob die Spieler-ID einen Spielerplatz in der Spielsitzung reserviert hat. Nach der Bestätigung ändert Amazon GameLift Servers den Status des Spielerplatzes von RESERVED nach ACTIVE.

Syntax

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

Parameters

playerSessionId

Eindeutige ID, die vom Amazon GameLift Servers Dienst als Antwort auf einen Aufruf der SDK-API-Aktion ausgegeben wird AWS . Amazon GameLift Servers [CreatePlayerSession](#) Der Spielclient verweist auf diese ID, wenn er eine Verbindung zum Serverprozess herstellt.

Typ: FString

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

ActivateGameSession()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess eine Sitzungs-ID für das Spiel ist aktiviert hat und nun Verbindungen annehmen kann. Diese Aktion muss im Rahmen der

onStartGameSession() -Callback-Funktion aufgerufen werden, nachdem die Initialisierung der Spielsitzung vollständig abgeschlossen ist.

Syntax

```
FGameLiftGenericOutcome ActivateGameSession()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

DescribePlayerSessions()

Ruft Sitzungsdaten der Spieler, einschließlich Einstellungen, Metadaten zu der Sitzung und Spielerdaten ab. Verwenden Sie diese Aktion, um für eine einzelne Spilersitzung, für alle Spilersitzungen in einer Spielsitzung oder für alle Spilersitzungen zu einer einzelnen Spieler-ID abzurufen.

Syntax

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const  
FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

Parameters

describePlayerSessionsAnfrage

Ein [FDescribePlayerSessionsRequest](#)-Objekt, mit dem beschrieben wird, welche Spilersitzungen abgerufen werden sollen.

Erforderlich: Ja

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein [FDescribePlayerSessionsRequest](#)-Objekt mit einer Menge von Spilersitzungsobjekten zurück, die den Anforderungsparametern

entsprechen. Player-Sitzungsobjekte haben eine Struktur, die mit dem AWS Amazon GameLift Servers [PlayerSession](#) SDK-API-Datentyp identisch ist.

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die derzeit von dem Serverprozess gehostet wird, wenn der Serverprozess aktiv ist.

Syntax

```
FGameLiftStringOutcome GetGameSessionId()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als `FGameLiftStringOutcome`-Objekt zurück. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

GetInstanceCertificate()

Ruft den Dateispeicherort eines PEM-codierten TLS-Zertifikats ab, das der Flotte und ihren Instanzen zugeordnet ist. AWS Certificate Manager generiert dieses Zertifikat, wenn Sie eine neue Flotte erstellen, bei der die Zertifikatskonfiguration auf `GENERATED` gesetzt ist. Verwenden Sie dieses Zertifikat, um eine sichere Verbindung mit einem Spiele-Client herzustellen und die Client-/Serverkommunikation zu verschlüsseln.

Syntax

```
FGameLiftGetInstanceCertificateOutcome GetInstanceCertificate()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein `GetInstanceCertificateOutcome` Objekt zurückgegeben, das den Speicherort der TLS-Zertifikatsdatei und der Zertifikatskette der Flotte enthält, die auf der Instanz

gespeichert sind. Eine Stammzertifikatsdatei, die aus der Zertifikatskette extrahiert wurde, wird ebenfalls auf der Instanz gespeichert. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Weitere Informationen zu den Daten des Zertifikats und der Zertifikatskette finden Sie unter [GetCertificate Response Elements](#) in der AWS Certificate Manager API-Referenz.

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
FGameLiftStringOutcome GetSdkVersion();
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als `FGameLiftStringOutcome`-Objekt zurück. Die zurückgegebene Zeichenfolge enthält nur die Versionsnummer (z. B. „3.1.5“). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Initialisiert das Amazon GameLift Servers-SDK. Diese Methode sollte beim Start aufgerufen werden, bevor andere Amazon GameLift Servers-Initialisierungen durchgeführt werden.

Syntax

```
FGameLiftGenericOutcome InitSDK()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

ProcessEnding()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess beendet wird. Diese Methode sollte aufgerufen werden, nachdem alle anderen Aufgaben ausgeführt sind, und insbesondere auch alle aktiven Spielsitzungen beendet sind. Diese Methode sollte mit einem Beendigungscode 0 beendet werden. Andere BeendigungsCodes sorgen für eine Ereignisnachricht zum unsauberen Beenden des Prozesses.

Syntax

```
FGameLiftGenericOutcome ProcessEnding()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

ProcessReady()

Benachrichtigt den Amazon GameLift Servers-Service, dass der Serverprozess zum Hosten von Spielsitzungen bereit ist. Rufen Sie diese Methode auf, nachdem Sie erfolgreich Einrichtungsaufgaben aufgerufen [InitSDK\(\)](#) und abgeschlossen haben, die erforderlich sind, bevor der Serverprozess eine Spielsitzung hosten kann. Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
FGameLiftGenericOutcome ProcessReady(FProcessParameters &processParameters)
```

Parameters

FProcessParameter

Ein [FProcessParameter](#)-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Die Namen von Callback-Methoden, implementiert in dem Code des Spiel-Servers, den der Amazon GameLift Servers-Service zur Kommunikation mit dem Serverprozess aufruft.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu beliebigen Spielsitzungsspezifischen Dateien, die Amazon GameLift Servers erfassen und speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

Vgl. den Beispielcode in [Verwendung des Unreal Engine-Plug-ins](#).

RemovePlayerSession()

Benachrichtigt den Amazon GameLift Servers-Service, dass ein Spieler mit der angegebenen Sitzungs-ID eine Verbindung zu dem Server getrennt hat. Als Reaktion ändert Amazon GameLift Servers den Status des Spielerplatzes in „verfügbar“, so dass er einem neuen Spieler zugewiesen werden kann.

Syntax

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerId)
```

Parameters

playerSessionId

Eindeutige ID, die vom Amazon GameLift Servers Dienst als Antwort auf einen Aufruf der AWS Amazon GameLift Servers SDK-API-Aktion ausgegeben wurde [CreatePlayerSession](#). Der Spielclient verweist auf diese ID, wenn er eine Verbindung zum Serverprozess herstellt.

Typ: FString

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Siehe auch die AWS SDK-Aktion [StartMatchBackfill\(\)](#). Mit dieser Aktion können Match-Backfill-Anforderungen von einem Game-Server-Prozess initiiert werden, der die Spielsitzung hostet. Erfahren Sie mehr über die [FlexMatchBackfill-Funktion](#).

Diese Aktion ist asynchron. Wenn neue Spieler erfolgreich gematcht werden, liefert der Amazon GameLift Servers-Service über die Callback-Funktion `OnUpdateGameSession()` aktualisierte Matchmaker-Daten.

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst [StopMatchBackfill\(\)](#) auf, um die ursprüngliche Anforderung abubrechen.

Syntax

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest  
&startBackfillRequest);
```

Parameters

FStartMatchBackfillRequest

Ein [FStartMatchBackfillRequest](#)-Objekt, das die folgenden Informationen übermittelt:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Informationen sind optional. Bei fehlender ID-Angabe wird automatisch eine ID von Amazon GameLift Servers generiert.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert kann aus den Matchmaker-Daten der Spielsitzung abgerufen werden.

- Die ID der Spielsitzung für das Backfill.
- Verfügbare Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Erforderlich: Ja

Rückgabewert

Wenn erfolgreich, wird das Match-Backfill-Ticket als `FGameLiftStringOutcome`-Objekt zurückgegeben. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben. Der Ticketstatus kann mit der AWS SDK-Aktion [DescribeMatchmaking\(\)](#) verfolgt werden.

StopMatchBackfill()

Bricht eine aktive mit dem Befehl [StartMatchBackfill\(\)](#) erstellte Match-Backfill-Anforderung ab. Siehe auch die AWS SDK-Aktion [StopMatchmaking\(\)](#). Erfahren Sie mehr über die [FlexMatchBackfill-Funktion](#).

Syntax

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest  
&stopBackfillRequest);
```

Parameters

StopMatchBackfillRequest

Ein [FStopMatchBackfillRequest](#)-Objekt, das das abzubrechende Matchmaking-Ticket identifiziert:

- Ticket-ID der abzubrechenden Backfill-Anforderung
- Matchmaker, an den die Backfill-Anforderung gesendet wurde
- Spielsitzung für die Backfill-Anforderung

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

TerminateGameSession()

Diese Methode ist mit Version 4.0.1 veraltet. Stattdessen sollte der Serverprozess [ProcessEnding\(\)](#) nach dem Ende einer Spielsitzung aufgerufen werden.

Informiert den Amazon GameLift Servers Dienst darüber, dass der Serverprozess die aktuelle Spielsitzung beendet hat. Diese Aktion wird aufgerufen, wenn der Serverprozess aktiv bleibt und bereit ist, eine neue Spielsitzung zu veranstalten. Sie sollte erst aufgerufen werden, nachdem der Vorgang zum Beenden der Spielsitzung abgeschlossen ist, da sie signalisiert Amazon GameLift Servers, dass der Serverprozess sofort für das Hosten einer neuen Spielsitzung verfügbar ist.

Diese Aktion wird nicht aufgerufen, wenn der Serverprozess nach dem Ende der Spielsitzung heruntergefahren wird. Rufen Sie stattdessen auf, [ProcessEnding\(\)](#) um zu signalisieren, dass sowohl die Spielsitzung als auch der Serverprozess beendet werden.

Syntax

```
FGameLiftGenericOutcome TerminateGameSession()
```

Parameters

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg besteht, mit einer Fehlermeldung.

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spiellersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt. (Siehe auch die Aktion [UpdateGameSession\(\)](#) in der Amazon GameLift Servers-Service API-Referenz).

Syntax

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy policy)
```

Parameters

Richtlinie

Wert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Eingabe: `EPlayerSessionCreationPolicy` enum. Gültige Werte sind:

- `ACCEPT_ALL` – Akzeptiert alle neuen Spielsitzungen.
- `DENY_ALL` – Verwehrt neue Spielsitzungen.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Platzierungsveranstaltungen für Spielsitzungen

Amazon GameLift Servers sendet Ereignisse für jede Anfrage zur Platzierung einer Spielsitzung aus, während diese verarbeitet wird. Sie können diese Ereignisse in einem Amazon SNS SNS-Thema veröffentlichen, wie unter beschrieben [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#). Diese Ereignisse werden auch nahezu in Echtzeit und nach bestem Wissen und Gewissen an Amazon CloudWatch Events gesendet.

Dieses Thema beschreibt die Struktur der Platzierungsveranstaltungen für Spielsitzungen und bietet ein Beispiel für jeden Ereignistyp. Weitere Informationen zum Status von Anfragen zur Platzierung von Spielsitzungen finden Sie [GameSessionPlacement](#) im Amazon GameLift Servers API-Referenz.

Syntax für Platzierungsereignis

Ereignisse werden als JSON-Objekte dargestellt. Die Struktur der CloudWatch Ereignisse entspricht dem Ereignismuster mit ähnlichen Feldern auf oberster Ebene und dienstspezifischen Details.

Zu den Feldern der obersten Ebene gehören die folgenden Felder (weitere Informationen finden Sie unter [Ereignismuster](#)):

version

Dieses Feld ist immer auf 0 (Null) gesetzt.

id

Eindeutige Tracking-ID für das Ereignis.

detail-type

Wert ist immer `GameLift Queue Placement Event`.

Quelle

Wert ist immer `aws.gamelift`.

Konto

Das AWS Konto, das zur Verwaltung verwendet wird Amazon GameLift Servers.

time

Zeitstempel des Ereignisses.

Region

Die AWS Region, in der die Platzierungsanfrage bearbeitet wird. Dies ist die Region, in der sich die aktuell verwendete Warteschlange für Spielsitzungen befindet.

Ressourcen

ARN-Wert der Warteschlange für die Spielsitzung, die die Platzierungsanfrage verarbeitet.

PlacementFulfilled

Die Platzierungsanfrage wurde erfolgreich erfüllt. Eine neue Spielsitzung wurde gestartet und für jeden Spieler, der in der Platzierungsanfrage für die Spielsitzung aufgeführt ist, wurden neue Spielsitzungen erstellt. Verbindungsinformationen für Spieler sind verfügbar.

Syntax im Detail:

Platzierungs-ID

Eine eindeutige Kennung, die der Platzierungsanfrage für die Spielsitzung zugewiesen wurde.

port

Die Portnummer für die neue Spielsitzung.

gameSessionArn

Die ARN-ID für die neue Spielsitzung.

ipAddress

Die IP-Adresse der Spielsitzung.

DNS-Name

Die DNS-ID, die der Instanz zugewiesen wurde, auf der die neue Spielsitzung ausgeführt wird. Das Wertformat ist unterschiedlich, je nachdem, ob die Instanz, auf der die Spielsitzung ausgeführt wird, TLS-fähig ist. Wenn Spieler auf einer TLS-fähigen Flotte eine Verbindung zu einer Spielsitzung herstellen, müssen sie den DNS-Namen verwenden, nicht die IP-Adresse.

TLS-fähige Flotten: `<unique identifier>.<region identifier>.amazongamelift.com`

Non-TLS-enabled Flotten: `ec2-<unique identifier>.compute.amazonaws.com`

startTime

Zeitstempel, der angibt, wann diese Anfrage in die Warteschlange gestellt wurde.

endTime

Zeitstempel, der angibt, wann diese Anfrage erfüllt wurde.

gameSessionRegion

AWS Region der Flotte, die die Spielsitzung veranstaltet. Dies entspricht dem Regionstoken in der `GameSessionArn`.

placedPlayerSessions

Die Sammlung von Spielersitzungen, die für jeden Spieler in der Platzierungsanfrage für die Spielsitzung erstellt wurden.

Beispiel

```
{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
```

```

    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {
    "type": "PlacementFulfilled",
    "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
    "port": "6262",
    "gameSessionArn": "arn:aws:gamelift:us-west-2::gamesession/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa/4444dddd-55ee-66ff-77aa-8888bbbb99cc",
    "ipAddress": "98.987.98.987",
    "dnsName": "ec2-12-345-67-890.us-west-2.compute.amazonaws.com",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z",
    "gameSessionRegion": "us-west-2",
    "placedPlayerSessions": [
      {
        "playerId": "player-1"
        "playerSessionId": "psess-1232131232324124123123"
      }
    ]
  }
}

```

PlacementCancelled

Die Platzierungsanfrage wurde mit einem Anruf beim GameLift Service storniert
[StopGameSessionPlacement](#).

Detail:

Platzierungs-ID

Eine eindeutige Kennung, die der Platzierungsanfrage für die Spielsitzung zugewiesen wurde.

startTime

Zeitstempel, der angibt, wann diese Anfrage in die Warteschlange gestellt wurde.

endTime

Zeitstempel, der angibt, wann diese Anfrage storniert wurde.

Beispiel

```
{
```

```
"version": "0",
"id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
"detail-type": "GameLift Queue Placement Event",
"source": "aws.gamelift",
"account": "123456789012",
"time": "2021-03-01T15:50:52Z",
"region": "us-east-1",
"resources": [
  "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
],
"detail": {

  "type": "PlacementCancelled",
  "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
  "startTime": "2021-03-01T15:50:49.741Z",
  "endTime": "2021-03-01T15:50:52.084Z"
}
}
```

PlacementTimedOut

Die Platzierung der Spielsitzung wurde nicht erfolgreich abgeschlossen, bevor das Zeitlimit der Warteschlange abgelaufen ist. Die Platzierungsanfrage kann bei Bedarf erneut eingereicht werden.

Detail:

Platzierungs-ID

Eine eindeutige Kennung, die der Platzierungsanfrage für die Spielsitzung zugewiesen wurde.

startTime

Zeitstempel, der angibt, wann diese Anfrage in die Warteschlange gestellt wurde.

endTime

Zeitstempel, der angibt, wann diese Anfrage storniert wurde.

Beispiel

```
{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
```

```
"detail-type": "GameLift Queue Placement Event",
"source": "aws.gamelift",
"account": "123456789012",
"time": "2021-03-01T15:50:52Z",
"region": "us-east-1",
"resources": [
  "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
],
"detail": {

  "type": "PlacementTimedOut",
  "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
  "startTime": "2021-03-01T15:50:49.741Z",
  "endTime": "2021-03-01T15:50:52.084Z"
}
}
```

PlacementFailed

Amazon GameLift Servers konnte die Anfrage für die Spielsitzung nicht erfüllen. Dies wird in der Regel durch einen unerwarteten internen Fehler verursacht. Die Platzierungsanfrage kann bei Bedarf erneut eingereicht werden.

Detail:

Platzierungs-ID

Eine eindeutige Kennung, die der Platzierungsanfrage für die Spielsitzung zugewiesen wurde.

startTime

Zeitstempel, der angibt, wann diese Anfrage in die Warteschlange gestellt wurde.

endTime

Zeitstempel, der angibt, wann diese Anfrage fehlgeschlagen ist.

Beispiel

```
{
  "version": "0",
  "id": "39c978f3-ba46-3f7c-e787-55bfcca1bd31",
  "detail-type": "GameLift Queue Placement Event",
```



```

"source": "aws.gamelift",
"account": "252386620677",
"time": "2021-03-01T15:50:52Z",
"region": "us-east-1",
"resources": [
  "arn:aws:gamelift:us-west-2:252386620677:gamesessionqueue/MegaFrogRace-NA"
],
"detail": {

  "type": "PlacementFailed",
  "placementId": "e4a1119a-39af-45cf-a990-ef150fe0d453",
  "startTime": "2021-03-01T15:50:49.741Z",
  "endTime": "2021-03-01T15:50:52.084Z"
}
}

```

Amazon GameLift ServersAMI-Versionen

In der folgenden Tabelle sind die neuesten Amazon-Maschinen-Images (AMIs) aufgeführt, die für verwaltetes EC2-Hosting Amazon GameLift Servers verwendet werden. Wie unter [beschrieben Konfigurations- und Schwachstellenanalyse in Amazon GameLift Servers](#), müssen Sie regelmäßig neue Amazon GameLift Servers verwaltete EC2-Flotten erstellen, um die neuesten AMI-Versionsupdates bereitzustellen.

AMIs zur Verwendung mit Server-SDK für 5+ Amazon GameLift Servers

Amazon GameLift Servers verwendet Folgendes AMIs, um Spieleserver zu hosten, die in das Server-SDK für Amazon GameLift Servers Version 5 integriert sind.

Amazon GameLift Servers-Abbild	Architektur	Neuester Patch	AWS Basisbild
Amazon Linux 2023 BASE_AMI_LINUX_2023	86 x	16.12.2025	Amazon Linux 2023 AMI 2023.9.20 251208.0 x86_64 HVM-Kernel-6.1 (Versionshinweise)
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_5	86 x	16.12.2025	Amazon Linux 2 Kernel 5.10 AMI 2.0.20251208.0 x86_64 HVM gp2 (Versionshinweise)

Amazon GameLift Servers-Abbild	Architektur	Neuester Patch	AWS Basisbild
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_5	86 x	16.12.2025	Windows_Server-2016-Englisch-Full-Base-2025.12.10 (Versionshinweise)
Amazon Linux 2023 BASE_AMI_LINUX_2023_ARM	ARM64	16.12.2025	Amazon Linux 2023 AMI 2023.9.20 251208.0 arm64 HVM Kernel-6.1 (Versionshinweise)
Amazon Linux 2 BASE_AMI_LINUX_2_ARM	ARM64	16.12.2025	Amazon Linux 2 LTS Arm64 Kernel 5.10 AMI 2.0.20251208.0 arm64 HVM gp2 (Versionshinweise)
Windows 2022 BASE_AMI_WINDOWS_2022	86 x	16.12.2025	Windows_Server-2022-Englisch-Full-Base-2025.12.10 (Versionshinweise)

AMIs zur Verwendung mit dem Server-SDK für 4 Amazon GameLift Servers

Amazon GameLift Servers verwendet Folgendes AMIs, um Spieleserver zu hosten, die in das Server-SDK für Amazon GameLift Servers Version 4 oder früher integriert sind.

Amazon GameLift Servers-Abbild	Architektur	Neuester Patch	AWS Basisbild
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_4	86 x	16.12.2025	Amazon Linux 2 AMI 2.0.20251208.0 x86_64 HVM gp2 (Versionshinweise)
Amazon Linux 2023 BASE_AMI_LINUX_2023_SDK_4	86 x	2025-12-16	Amazon Linux 2023 AMI 2023.9.20 251208.0 x86_64 HVM-Kernel-6.1 (Versionshinweise)

Amazon GameLift Servers-Abbild	Architektur	Neuester Patch	AWS Basisbild
Nur in den Regionen China (Peking) und China (Ningxia) verfügbar.			
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_4	86 x	16.12.2025	Windows_Server-2016-Englisch-Full-Base-2025.12.10 (Versionshinweise)

Weitere Informationen finden Sie in den folgenden Ressourcen:

- [Versionshinweise zu Amazon Linux 2023](#)
- [Versionshinweise zu Amazon Linux 2](#)
- [AWS Windows AMI-Versionsverlauf](#)
- [Beschreibung der inhaltlichen Änderungen von Software Update Services und Windows Server Update Services für 2024](#)

Amazon GameLift Servers Servicestandorte

Amazon GameLift Servers Funktionen sind in mehreren AWS-Regionen und Local Zones verfügbar. Sie können eine Hosting-Lösung entwerfen, die Ihre Spieleserver genau dort platziert, wo sich Ihre Spieler befinden.

Unterstützte AWS Standorte

Die folgende Tabelle enthält eine Liste der AWS-Regionen Local Zones, die Amazon GameLift Servers Ressourcen unterstützen. Sie gibt an, welche Ressourcentypen Sie an den einzelnen Standorten erstellen können.

Geografischer Standort	Ortscode	Heimatreion für verwaltete Flotten (einzelner Standort)	Heimatreion für verwaltete Flotten (mehrere Standorte)	Remote-Standort für verwaltete Flotten (mehrere Standorte)	Flotte an jedem Ort	Warteschlange für Spielsitzungen	FlexMatchmaker und Regelsatz	DDoS-Schutz (Spieler-Gateway)
USA Ost (Nord-Virginia)	us-east-1	Ja	Ja	Ja	Ja	Ja	Ja	Ja [†]
USA Ost (Ohio)	us-east-2	Ja		Ja	Ja	Ja		
USA West (Nordkalifornien)	us-west-1	Ja		Ja	Ja	Ja		
USA West (Oregon)	us-west-2	Ja	Ja	Ja	Ja	Ja	Ja	Ja [†]
Afrika (Kapstadt)	af-south-1			Ja				
Asien-Pazifik (Thailand)	ap-southeast-7			Ja				
Asien-Pazifik (Hongkong)	ap-east-1			Ja				
Asien-Pazifik (Malaysia)	ap-southeast-5			Ja				

Geografischer Standort	Ortscode	Heimatreion für verwaltete Flotten (einzelner Standort)	Heimatreion für verwaltete Flotten (mehrere Standorte)	Remote-Standort für verwaltete Flotten (mehrere Standorte)	Flotte an jedem Ort	Warteschlange für Spielsitzungen	FlexMatchmaker und Regelsatz	DDoS-Schutz (Spieler-Gateway)
Asien-Pazifik (Mumbai)	ap-south-1	Ja		Ja	Ja	Ja		
Asien-Pazifik (Osaka)	ap-northeast-3			Ja				
Asien-Pazifik (Seoul)	ap-northeast-2	Ja	Ja	Ja	Ja	Ja	Ja	Ja [†]
Asien-Pazifik (Singapur)	ap-southeast-1	Ja		Ja	Ja	Ja		
Asien-Pazifik (Sydney)	ap-southeast-2	Ja	Ja	Ja	Ja	Ja	Ja	Ja [†]
Asien-Pazifik (Tokio)	ap-northeast-1	Ja	Ja	Ja	Ja	Ja	Ja	Ja [†]
Kanada (Zentral)	ca-central-1	Ja		Ja	Ja	Ja		

Geografischer Standort	Ortscode	Heimatreion für verwaltete Flotten (einzelner Standort)	Heimatreion für verwaltete Flotten (mehrere Standorte)	Remote-Standort für verwaltete Flotten (mehrere Standorte)	Flotte an jedem Ort	Warteschlange für Spielsitzungen	FlexMatchmaker und Regelsatz	DDoS-Schutz (Spieler-Gateway)
Europa (Frankfurt)	eu-central-1	Ja	Ja	Ja	Ja	Ja	Ja	Ja [†]
Europa (Irland)	eu-west-1	Ja	Ja	Ja	Ja	Ja	Ja	Ja [†]
Europa (London)	eu-west-2	Ja		Ja	Ja	Ja		
Europa (Milan)	eu-south-1			Ja				
Europa (Paris)	eu-west-3			Ja				
Europa (Stockholm)	eu-north-1			Ja				
Naher Osten (Bahrain)	me-south-1			Ja				
Südamerika (São Paulo)	sa-east-1	Ja		Ja	Ja	Ja		
Lokale Zone von Atlanta	us-east-1-atl-1			Ja				

Geografischer Standort	Ortscode	Heimatreion für verwaltete Flotten (einzelner Standort)	Heimatreion für verwaltete Flotten (mehrere Standorte)	Remote-Standort für verwaltete Flotten (mehrere Standorte)	Flotte an jedem Ort	Warteschlange für Spielsitzungen	FlexMatchmaker und Regelsatz	DDoS-Schutz (Spieler-Gateway)
Lokale Zone von Chicago	us-ost-1-chi-1			Ja				
Lokale Zone von Dallas *	us-ost-1-dfw-1			Ja				
Lokale Zone 2 von Dallas	us-east-1-dfw-2			Ja				
Lokale Zone von Denver	us-west-2-den-1			Ja				
Lokale Zone von Houston	us-ost-1-iah-1			Ja				
Lokale Zone von Kansas City	us-east-1-mci-1			Ja				
Lokale Zone von Los Angeles	us-west-2-lax-1			Ja				

Geografischer Standort	Ortscode	Heimatregion für verwaltete Flotten (einzelner Standort)	Heimatregion für verwaltete Flotten (mehrere Standorte)	Remote-Standort für verwaltete Flotten (mehrere Standorte)	Flotte an jedem Ort	Warteschlange für Spielsitzungen	FlexMatchmaker und Regelsatz	DDoS-Schutz (Spieler-Gateway)
Lokale Zone von Phoenix	us-west-2-Phx-1			Ja				
Lagos, Nigeria, lokale Zone	af-south-1-los-1			Ja				

* Verfügbar für diejenigen AWS-Konten, die sich bereits angemeldet haben.

† Gültige Fleet-Home-Region für DDoS-Schutz (Spieler-Gateway).

Note

Einige AWS-Regionen und alle Local Zones sind standardmäßig für eine nicht aktivierte AWS-Konto. Du musst dich zuerst für diese Regionen oder Local Zones anmelden, bevor du Spieleserver an diesen Standorten einsetzen kannst. Weitere Informationen zu Regionen, die standardmäßig nicht aktiviert sind, und wie du sie aktivierst, findest du unter [Verwaltung AWS-Regionen](#) in der Allgemeinen AWS-Referenz. Weitere Informationen finden Sie unter [Erste Schritte mit Local Zones](#) im AWS Local Zones Zones-Benutzerhandbuch. (Flotten, die vor dem 28. Februar 2022 erstellt wurden, sind von dieser Anforderung nicht betroffen.) Darüber hinaus müssen Sie Ihre Amazon GameLift Servers Administratorrichtlinie aktualisieren, um die `ec2:DescribeRegions` Aktion zuzulassen. Ein Richtlinienbeispiel mit Regionen, die nicht standardmäßig aktiviert sind, finden Sie unter [Beispiele für Administratorberechtigungen](#).

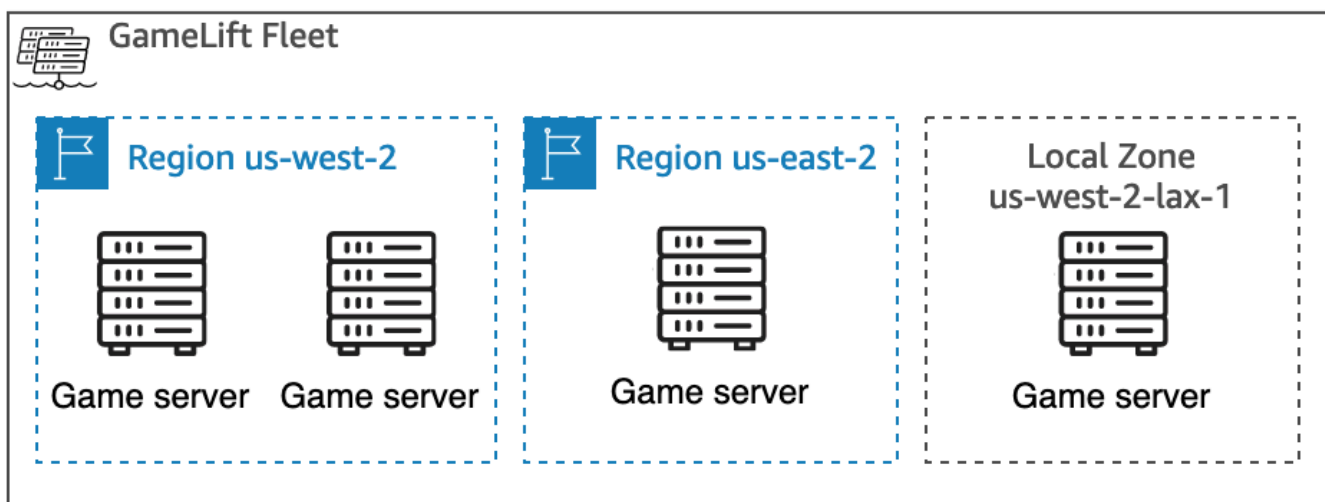
Standorte für verwaltetes Hosting

Amazon GameLift Servers Beim verwalteten Hosting werden Flotten von Spielserverressourcen bereitgestellt. Jede Flotte wird in einer AWS-Region, der Heimatregion der Flotte, erstellt. Auf die Heimatregion einer Flotte wird in der Amazon-Ressourcennummer (ARN) der Flotte verwiesen.

Sie können eine Flotte mit einer einzigen Region einsetzen und Hosting-Ressourcen nur in der Heimatregion bereitstellen. Alternativ können Sie eine Flotte mit mehreren Standorten bereitstellen, wobei Hosting-Ressourcen an mehreren geografischen Standorten bereitgestellt werden. Eine Flotte mit mehreren Standorten hat eine Heimatregion und einen oder mehrere abgelegene Standorte. Es gibt keine Einschränkungen bei der Gruppierung von Standorten in einer Flotte. Bei der Verwaltung der Hosting-Kapazität für eine Flotte mit mehreren Standorten können Sie die Kapazität für jeden Standort einzeln festlegen.

Abgelegene Standorte für eine Flotte mit mehreren Standorten können andere AWS-Regionen oder Local Zones sein. Eine lokale Zone ist eine Erweiterung einer AWS-Region. Damit können Sie Rechenressourcen näher an den Benutzern platzieren, um ein Gameplay mit geringerer Latenz zu ermöglichen. Weitere Informationen finden Sie unter [AWS Local Zones](#). Der Standortcode für eine lokale Zone ist der Code der übergeordneten Region, gefolgt von einer physischen Standortkennung. Der Code für die lokale Zone Los Angeles lautet beispielsweise `us-west-2-lax-1`.

Das folgende Diagramm zeigt eine Flotte mit mehreren Standorten und Ressourcen in zwei AWS-Regionen und einer lokalen Zone. Die Heimatregion der Flotte ist `us-west-2`, und sie hat zwei abgelegene Standorte: `us-east-2` Region und `us-west-2-lax-1` Lokale Zone.



Zusätzlich zu den Flottenressourcen Amazon GameLift Servers kann Managed Hosting mit auch die folgenden Arten von Ressourcen verwenden. Sie erstellen diese Ressourcen in einem bestimmten Format AWS-Region , das den Ressourcentyp unterstützt.

- **Build** — Dies ist ein Spielservers-Build, der mit einer verwalteten EC2-Flotte gehostet wird. Erstelle eine Build-Ressource in derselben Region wie die Flotte, in der sie eingesetzt werden soll.
- **Skript** — Dies ist ein Konfigurationsskript für das Hosten eines Spiels mit Amazon GameLift ServersRealtime. Erstellen Sie eine Skriptressource in derselben Region wie die Flotte, für die sie eingesetzt werden soll.
- **Container-Gruppendefinition** — Dies ist eine Konfiguration für den Betrieb von Containern auf einer verwalteten Containerflotte. Es identifiziert ein oder mehrere Container-Images mit Gameserver-Software, die in der Containerflotte bereitgestellt werden sollen. Erstellen Sie eine Container-Gruppendefinition in derselben Region wie die Flotte, in der sie eingesetzt werden sollen.
- **Warteschlange für Spielsitzungen** — Diese Ressource verarbeitet Anfragen für Spielsitzungen und initiiert neue Spielsitzungen. Die Verarbeitung findet dort statt AWS-Region , wo sich die Warteschlange befindet. Um die Latenz bei der Platzierung von Spielsitzungen zu verringern, erstellen Sie eine Warteschlange, die sich geografisch in der Nähe der Spieler befindet, die sie verwenden werden.

Standorte für überall Amazon GameLift Servers

Eine Amazon GameLift Servers Anywhere-Flotte ist eine Sammlung von Hosting-Hardware, die Sie bereitstellen. Sie verwalten alle Aktivitäten auf Ihren Hosting-Ressourcen, einschließlich der Bereitstellung von Gameserver-Software, deren Aktualisierung und dem Starten von Serverprozessen. Sie erstellen eine Anywhere-Flotte, um den Amazon GameLift Servers Service mit Ihren selbst verwalteten Hosting-Ressourcen zu verbinden. Amazon GameLift Servers verwaltet die Platzierung von Spielsitzungen — verarbeitet Anfragen zum Beitritt von Spielern, sucht nach verfügbaren Hosting-Ressourcen, initiiert neue Spielsitzungen und stellt den Spielclients Verbindungsinformationen zur Verfügung. Sie können eine Anywhere-Flotte in einer beliebigen Flotte erstellen, die AWS-Regionen sie unterstützt.

Sie fügen Instanzen von Hosting-Hardware zu einer Anywhere-Flotte hinzu, indem Sie sie registrieren. Jeder registrierten Instanz muss ein benutzerdefinierter Standort zugeordnet sein. Benutzerdefinierte Standorte haben nichts mit Local Zones zu AWS-Regionen tun. Sie werden verwendet, um den physischen Standort der Hardware darzustellen.

Weitere Informationen zum Erstellen einer Anywhere-Flotte und zum Testen der Gameserver-Integration finden Sie unter [Erstelle eine Amazon GameLift Servers Überall Flotte](#) und [Lokales Testen mit Amazon GameLift Servers Anywhere einrichten](#).

Standorte für Amazon GameLift Servers FlexMatch

FlexMatchRessourcen werden verwendet, um Spieleranfragen für das Matchmaking zu bearbeiten. Sie umfassen eine Matchmaking-Konfigurationsressource und eine Regelsatzressource. Die Verarbeitung erfolgt AWS-Region dort, wo sich die FlexMatch Ressourcen befinden. Um die Latenz beim Matchmaking-Prozess zu verringern, sollten Sie die Ressourcen geografisch in der Nähe der Spieler platzieren, die sie verwenden werden. Eine Matchmaking-Konfiguration und der Regelsatz, den sie verwendet, müssen sich in derselben befinden. AWS-Region Sie können in allen, die sie unterstützen AWS-Regionen , FlexMatch Ressourcen erstellen.

Weitere Informationen FlexMatch zur Einrichtung Ihrer Hosting-Lösung finden Sie im [Amazon GameLift ServersFlexMatchEntwicklerhandbuch](#).

Amazon GameLift Serversin China

Wenn Sie Ressourcen in der Region China (Peking), betrieben von Sinnet, oder in der Region China (Ningxia), betrieben von NWCD, verwenden Amazon GameLift Servers möchten, benötigen Sie ein separates Konto AWS (China). Beachten Sie, dass einige Funktionen in den Regionen China nicht verfügbar sind. Weitere Informationen zur Verwendung Amazon GameLift Servers in diesen Regionen finden Sie in den folgenden Ressourcen:

- [Amazon Web Services in China](#)
- [Amazon GameLift Servers](#)(Erste Schritte mit Amazon Web Services in China)

Endpunkte und Kontingente von Amazon GameLift Servers

- [Dienstendpunkte und Beacon-Endpunkte, mit denen Sie programmgesteuert eine Verbindung mit dem Dienst herstellen können, finden Sie unter Dienstendpunkte. Amazon GameLift Servers](#)
- [Amazon GameLift ServersKontingente für die Nutzung von Dienstressourcen oder Vorgängen pro AWS Konto finden Sie unter Dienstkontingente. Amazon GameLift Servers](#) Weitere Informationen zu Kontingenten und dazu, wie Sie eine Erhöhung beantragen können, finden Sie unter [AWS Servicekontingenten](#). Sie können Kontingenterhöhungen über die Amazon GameLift Servers Konsole beantragen.

API-Limits für Amazon GameLift Servers

In den Tabellen in diesem Abschnitt sind die Standard-Ratenlimits für Amazon GameLift Server-API-Operationen aufgeführt. Diese Grenzwerte tragen zur Aufrechterhaltung der Servicestabilität und zur Gewährleistung einer fairen Nutzung bei. Jeder API-Vorgang hat spezifische Beschränkungen, die auf seinen Ressourcenauswirkungen und Nutzungsmustern basieren.

Standardmäßig ist das Drosselungslimit für jedes Kundenkonto auf 500 für Burst-Anfragen und 500 für normalen Traffic festgelegt. Darüber hinaus hat jede API ihr eigenes Standard-Limit-Paar, wie unten beschrieben.

Referenz zu API-Grenzwerten

In der folgenden Tabelle sind die Standard-Ratenlimits für Amazon GameLift Server-API-Operationen aufgeführt. APIs Wenn diese Option mit Drosselung auf Ressourcenebene gekennzeichnet ist, können zusätzliche Beschränkungen gelten, um Probleme mit Datenbank-Tastenkürzeln zu vermeiden.

Referenz zu API-Limits

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
AcceptMatch	100	100	Nein	Das Limit gilt pro Konto.
ClaimGameServer	40	20	Nein	Das Limit gilt pro Konto.
CreateAlias	4	1	Nein	Das Limit gilt pro Konto.
CreateBuild	10	1	Nein	Das Limit gilt pro Konto.
CreateContainerFleet	4	1	Nein	Das Limit gilt pro Konto.
CreateContainerGroupDefinition	10	1	Nein	Das Limit gilt pro Konto.
CreateFleet	4	1	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
CreateFleetLocations	4	1	Nein	Das Limit gilt pro Konto.
CreateGameServerGroup	5	0.1	Nein	Das Limit gilt pro Konto.
CreateGameSession	50	50	Nein	Das Limit gilt pro Konto.
CreateGameSessionQueue	4	1	Nein	Das Limit gilt pro Konto.
CreateLocation	4	1	Nein	Das Limit gilt pro Konto.
CreateMatchmakingConfiguration	4	1	Nein	Das Limit gilt pro Konto.
CreateMatchmakingRuleSet	4	1	Nein	Das Limit gilt pro Konto.
CreatePlayerSession	200	200	Nein	Das Limit gilt pro Konto.
CreatePlayerSessions	200	200	Nein	Das Limit gilt pro Konto.
CreateScript	10	1	Nein	Das Limit gilt pro Konto.
CreateVpcPeeringAuthorization	1	0.1	Nein	Das Limit gilt pro Konto.
CreateVpcPeeringConnection	1	0.1	Nein	Das Limit gilt pro Konto.
DeleteAlias	4	1	Nein	Das Limit gilt pro Konto.
DeleteBuild	10	1	Nein	Das Limit gilt pro Konto.
DeleteContainerFleet	4	1	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
DeleteContainerGroupDefinition	10	1	Nein	Das Limit gilt pro Konto.
DeleteFleet	4	1	Nein	Das Limit gilt pro Konto.
DeleteFleetLocations	4	1	Nein	Das Limit gilt pro Konto.
DeleteGameServerGroup	5	0.1	Nein	Das Limit gilt pro Konto.
DeleteGameSessionQueue	4	1	Nein	Das Limit gilt pro Konto.
DeleteLocation	4	1	Nein	Das Limit gilt pro Konto.
DeleteMatchmakingConfiguration	4	1	Nein	Das Limit gilt pro Konto.
DeleteMatchmakingRuleSet	4	1	Nein	Das Limit gilt pro Konto.
DeleteScalingPolicy	4	1	Nein	Das Limit gilt pro Konto.
DeleteScript	10	1	Nein	Das Limit gilt pro Konto.
DeleteVpcPeeringAuthorization	1	0.1	Nein	Das Limit gilt pro Konto.
DeleteVpcPeeringConnection	1	0.1	Nein	Das Limit gilt pro Konto.
DeregisterGameServer	40	20	Nein	Das Limit gilt pro Konto.
DeregisterCompute	10	2	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
DescribeAlias	10	2	Nein	Das Limit gilt pro Konto.
DescribeBuild	20	5	Nein	Das Limit gilt pro Konto.
DescribeCompute	10	2	Nein	Das Limit gilt pro Konto.
DescribeContainerFleet	20	5	Nein	Das Limit gilt pro Konto.
DescribeContainerGroupDefinition	20	5	Nein	Das Limit gilt pro Konto.
Beschreiben EC2 InstanceLimits	40	15	Nein	Das Limit gilt pro Konto.
DescribeFleetAttributes	10	2	Nein	Das Limit gilt pro Konto.
DescribeFleetCapacity	4	1	Nein	Das Limit gilt pro Konto.
DescribeFleetDeployment	20	5	Nein	Das Limit gilt pro Konto.
DescribeFleetEvents	10	5	Nein	Das Limit gilt pro Konto.
DescribeFleetLocationAttributes	10	2	Nein	Das Limit gilt pro Konto.
DescribeFleetLocationCapacity	10	5	Nein	Das Limit gilt pro Konto.
DescribeFleetLocationUtilization	10	2	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
DescribeFleetPortSettings	10	2	Nein	Das Limit gilt pro Konto.
DescribeFleetUtilization	10	2	Nein	Das Limit gilt pro Konto.
DescribeGameServer	40	20	Nein	Das Limit gilt pro Konto.
DescribeGameServerGroup	10	2	Nein	Das Limit gilt pro Konto.
DescribeGameServerInstances	10	2	Nein	Das Limit gilt pro Konto.
DescribeGameSessionDetails	10	5	Nein	Das Limit gilt pro Konto. Erwägen Sie DescribeGameSessions stattdessen die Verwendung, da es ein höheres Standardlimit hat und Limits auf Ressourcenebene unterstützt.
DescribeGameSessionPlacement	75	75	Ja	Das Limit gilt pro Konto. Vorbehaltlich einer Drosselung auf Ressourcenebene.
DescribeGameSessionQueues	10	2	Nein	Das Limit gilt pro Konto.
DescribeGameSessions	20	20	Ja	Das Limit gilt pro Konto. Vorbehaltlich einer Drosselung auf Ressourcenebene.
DescribeInstances	10	2	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
DescribeMatchmaking	200	200	Ja	Das Limit gilt pro Konto. Vorbehaltlich einer Drosselung auf Ressourcenebene.
DescribeMatchmakingConfigurations	15	7	Nein	Das Limit gilt pro Konto.
DescribeMatchmakingRuleSets	20	10	Nein	Das Limit gilt pro Konto.
DescribePlayerSessions	200	200	Ja	Das Limit gilt pro Konto. Vorbehaltlich einer Drosselung auf Ressourcenebene.
DescribeRuntimeConfiguration	4	1	Nein	Das Limit gilt pro Konto.
DescribeScalingPolicies	20	20	Nein	Das Limit gilt pro Konto.
DescribeScript	20	5	Nein	Das Limit gilt pro Konto.
DescribeVpcPeeringAuthorizations	1	0.1	Nein	Das Limit gilt pro Konto.
DescribeVpcPeeringConnections	1	0.1	Nein	Das Limit gilt pro Konto.
GetComputeAccess	4	1	Nein	Das Limit gilt pro Konto.
GetComputeAuthToken	10	5	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
GetGameSessionLogUrl	4	1	Nein	Das Limit gilt pro Konto.
GetInstanceAccess	4	1	Nein	Das Limit gilt pro Konto.
ListAliases	10	2	Nein	Das Limit gilt pro Konto.
ListBuilds	20	5	Nein	Das Limit gilt pro Konto.
ListContainerFleets	20	5	Nein	Das Limit gilt pro Konto.
ListCompute	10	2	Nein	Das Limit gilt pro Konto.
ListContainerGroupDefinitions	20	5	Nein	Das Limit gilt pro Konto.
ListContainerGroupDefinitionVersions	20	5	Nein	Das Limit gilt pro Konto.
ListFleets	10	2	Nein	Das Limit gilt pro Konto.
ListFleetDeployments	20	5	Nein	Das Limit gilt pro Konto.
ListGameServerGroups	10	2	Nein	Das Limit gilt pro Konto.
ListGameServers	10	2	Nein	Das Limit gilt pro Konto.
ListLocations	10	2	Nein	Das Limit gilt pro Konto.
ListScripts	20	5	Nein	Das Limit gilt pro Konto.
ListTagsForResource	20	5	Nein	Das Limit gilt pro Konto.
PutScalingPolicy	4	1	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
RegisterCompute	10	5	Nein	Das Limit gilt pro Konto.
RegisterGameServer	40	20	Nein	Das Limit gilt pro Konto.
RequestUploadCredentials	4	1	Nein	Das Limit gilt pro Konto.
ResolveAlias	4	1	Nein	Das Limit gilt pro Konto.
ResumeGameServerGroup	1	0.1	Nein	Das Limit gilt pro Konto.
SearchGameSessions	10	5	Nein	Das Limit gilt pro Konto.
StartFleetActions	1	0.1	Nein	Das Limit gilt pro Konto.
StartGameSessionPlacement	75	75	Nein	Das Limit gilt pro Konto.
StartMatchBackfill	70	70	Nein	Das Limit gilt pro Konto.
StartMatchmaking	100	100	Nein	Das Limit gilt pro Konto.
StopFleetActions	1	0.1	Nein	Das Limit gilt pro Konto.
StopGameSessionPlacement	20	20	Nein	Das Limit gilt pro Konto.
StopMatchmaking	100	100	Nein	Das Limit gilt pro Konto.
SuspendGameServerGroup	1	0.1	Nein	Das Limit gilt pro Konto.
TagResource	20	5	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
TerminateGameSession	10	2	Nein	Das Limit gilt pro Konto.
UntagResource	20	5	Nein	Das Limit gilt pro Konto.
UpdateAlias	4	1	Nein	Das Limit gilt pro Konto.
UpdateBuild	10	1	Nein	Das Limit gilt pro Konto.
UpdateContainerGroupDefinition	10	1	Nein	Das Limit gilt pro Konto.
UpdateContainerFleet	4	1	Nein	Das Limit gilt pro Konto.
UpdateFleetAttributes	4	1	Nein	Das Limit gilt pro Konto.
UpdateFleetCapacity	4	1	Nein	Das Limit gilt pro Konto.
UpdateFleetPortSettings	4	1	Nein	Das Limit gilt pro Konto.
UpdateGameServer	40	20	Nein	Das Limit gilt pro Konto.
UpdateGameServerGroup	5	0.1	Nein	Das Limit gilt pro Konto.
UpdateGameSession	10	2	Nein	Das Limit gilt pro Konto.
UpdateGameSessionQueue	4	1	Nein	Das Limit gilt pro Konto.
UpdateMatchmakingConfiguration	4	1	Nein	Das Limit gilt pro Konto.

API-Aktion	Burst-Limit	Ratenlimit	Drosselung auf Ressourcenebene	Hinweise
UpdateRuntimeConfiguration	4	1	Nein	Das Limit gilt pro Konto.
UpdateScript	10	1	Nein	Das Limit gilt pro Konto.
ValidateMatchmakingRuleSet	4	1	Nein	Das Limit gilt pro Konto.

Drosselung auf Ressourcenebene

Einige API-Operationen unterliegen einer Drosselung auf Ressourcenebene, um Probleme mit Datenbank-Hotkeys zu vermeiden. Die Limits auf API-Ebene für diese Operationen werden zwar erhöht, sie können aber dennoch auf Ressourcenebene gedrosselt werden.

Die folgenden API-Operationen unterliegen einer Drosselung auf Ressourcenebene:

- DescribeGameSessionPlacement
- DescribeGameSessions
- DescribeMatchmaking
- DescribePlayerSessions

Die Drosselung auf Ressourcenebene wird mithilfe bestimmter Drosseltasten erzwungen, die Ressourcenkennungen enthalten. DescribeGameSessionPlacement verwendet beispielsweise den Schlüssel „Operation:GameLift/,aws-account-placement-id:“DescribeGameSessionPlacement, der die Drosselung pro Kombination aus Konto und Platzierungs-ID erzwingt. Dadurch wird verhindert, dass Kunden das System mit Anfragen nach bestimmten Ressourcen überfordern.

API-Beschränkungen für Amazon GameLift Servers SDK

In der folgenden Tabelle sind die Standard-Ratenlimits für Amazon GameLift Servers SDK-API-Operationen aufgeführt. Diese Grenzwerte optimieren die Leistung und verhindern

Ressourcenkonflikte in Spieleserverumgebungen. Das Verständnis dieser Grenzwerte ist wichtig für eine effiziente serverseitige Integration mit Amazon GameLift Servern.

Server-SDK-Limits werden mithilfe bestimmter Drosselungsschlüssel durchgesetzt, die Konto- und Prozesskennungen enthalten. Einige Limits gelten pro Spieler, um Missbrauch durch einzelne Spieler zu verhindern.

Note

Diese Tabelle kann sich ändern. Die maßgebliche Quelle für aktuelle Grenzwerte ist die interne SDC-Konfiguration. Wenden Sie sich AWS bei Bedarf an den Support, um die meisten up-to-date Informationen zu erhalten.

Referenz zu den API-Limits des Server-SDK

API-Aktion	Burst-Limit	Ratenlimit	Drosselungstaste	Hinweise
AcceptPlayerSession	3	1	aws-account-and-process-id-and-player-id	Das Limit gilt pro Spieler und pro Prozess. Sollte niemals eine Erhöhung des Limits erforderlich sein, da es sich um eine Erhöhung pro Spieler handelt.
RemovePlayerSession	3	1	aws-account-and-process-id-and-player-id	Das Limit gilt pro Spieler und pro Prozess. Sollte niemals eine Erhöhung des Limits erforderlich sein, da es sich um eine Erhöhung pro Spieler handelt.
ActivateGameSession	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.

API-Aktion	Burst-Limit	Ratenlimit	Drosselungstaste	Hinweise
ActivateHostProcessV2	3	1	aws-account-and-process-ID	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
StartMatchBackfill	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
DescribePlayerSessions	10	5	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
GetComputeCertificate	3	0.1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
GetCustomRoleCredentials	3	0.1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
InitSDK	3	0.1	aws-account-and-process-ID	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.

API-Aktion	Burst-Limit	Ratenlimit	Drosselungstaste	Hinweise
RefreshHostProcess	3	0.1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
StopMatchmaking	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
ProcessEnding	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
ProcessReady	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
UpdateGameSession	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
UpdateHostProcess	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.

API-Aktion	Burst-Limit	Ratenlimit	Drosselungstaste	Hinweise
GetGameSessionId	10	5	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
GetTerminationTime	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.
UpdatePlayerSessionCreationPolicy	3	1	aws-account-and-process-id	Das Limit gilt pro Serverprozess. Eine Erhöhung des Limits sollte niemals erforderlich sein, da dies pro Prozess erfolgt.

Note

Dies sind die Standardgrenzwerte. Wenn Sie bei diesen eine Drosselung feststellen APIs, kann dies an der Drosselung auf Kontoebene liegen. Um eine Erhöhung des Limits zu beantragen, wenden Sie sich an den AWS Support.

Die Limits pro Prozess und pro Spieler sind so konzipiert, dass sie normalen Nutzungsmustern Rechnung tragen und sollten selten angepasst werden müssen. Wenn Sie bei diesen Systemen eine Drosselung feststellen APIs, überprüfen Sie Ihre Implementierung auf mögliche Optimierungsmöglichkeiten, bevor Sie eine Erhöhung der Limits beantragen. Die Drosselung auf Kontoebene kann auch aufgrund dieser zusätzlichen Regeln gelten:

- SdkWebSocketAccountLevelHighUsage/, aws-Konto
- SdkWebSocket/AccountLevelThrottle, aws-Konto

Amazon GameLift ServersUDP-Ping-Beacons

UDP-Ping-Beacons bieten eine Möglichkeit, die Netzwerklatenz zwischen Player-Geräten und Amazon GameLift Servers Hosting-Standorten zu messen. Mit Ping-Beacons können Sie genaue Latenzdaten sammeln, um fundierte Entscheidungen über die Platzierung von Spieleservern zu treffen und die Spielersuche auf der Grundlage der Latenzanforderungen zu verbessern.

Wie funktionieren UDP-Ping-Beacons

Amazon GameLift Servers bietet feste UDP-Endpunkte (Ping-Beacons) an jedem Hosting-Standort, an dem Sie Spieleserver bereitstellen können. Da die meisten Spieleserver über UDP kommunizieren, liefert die Messung der Latenz mit UDP-Ping-Beacons genauere Ergebnisse als die Verwendung von ICMP-Pings. Netzwerkgeräte verarbeiten ICMP-Pakete oft anders als UDP-Pakete, was zu Latenzmessungen führen kann, die nicht die tatsächliche Leistung widerspiegeln, die Ihre Spieler erleben werden.

Mit UDP-Ping-Beacons kann dein Spielclient UDP-Nachrichten an diese Endpunkte senden und asynchrone Antworten empfangen. So erhältst du Latenzmessungen, die die tatsächlichen Verkehrsbedingungen im Spiel zwischen dem Gerät eines Spielers und potenziellen Hosting-Standorten besser wiedergeben. Die Endpunkte sind permanent und bleiben verfügbar, solange das Hosten von Spielen an diesem Amazon GameLift Servers Standort unterstützt wird.

Häufige Anwendungsfälle für UDP-Ping-Beacons

Du kannst UDP-Ping-Beacons auf verschiedene Arten verwenden, um das Netzwerkerlebnis deines Spiels zu optimieren.

Auswahl optimaler Hosting-Standorte

Erfassen Sie Latenzdaten aus verschiedenen geografischen Regionen, um die besten Primär- und Backup-Standorte für das Hosten von Spieleservern für Ihre Spielerbasis zu ermitteln.

Platzieren von Spielsitzungen auf der Grundlage der Spielerlatenz

Füge Daten zur Spielerlatenz hinzu, wenn du neue Spielsitzungen anforderst, damit du Orte auswählen kannst, die die niedrigste Latenz bieten.

Optimieren Sie das Matchmaking auf der Grundlage der Latenz

Geben Sie Daten zur Spielerlatenz an, wenn Sie Matchmaking anfragen, um Spielern mit ähnlichen Latenzprofilen zu helfen und Spielsitzungen an optimalen Orten für passende Spieler zu platzieren.

 Note

Wenn du Anfragen zur Spielerzuweisung erstellst, solltest du keine Latenzinformationen an Standorte weitergeben, an denen du keine Flotten hast. Wenn ihr das tut, versucht Amazon GameLift Servers vielleicht, die Spielsitzung an Orten zu platzieren, an denen es keine Flottenkapazität gibt, was dazu führen kann, dass Spielerzuweisungsanfragen fehlschlagen.

Beacon-Endpunkte abrufen

Verwenden Sie den API-Vorgang, um Domänen- und Portinformationen für Ping-Beacons für Amazon GameLift Servers Standorte abzurufen. [ListLocations](#) Welche Speicherorte von dieser API zurückgegeben werden, hängt von den Speicherorten ab, die AWS-Region Sie beim Aufrufen der API angegeben haben (oder von Ihrer Standardregion, falls Sie keine angeben). Wenn Sie aufrufen von:

- Eine Heimatregion einer Flotte, die mehrere Standorte unterstützt: Die API gibt Informationen für alle Hosting-Standorte zurück
- Eine Heimatregion einer Flotte, die einen einzelnen Standort unterstützt: Die API gibt Informationen für diesen Standort zurück

Beachten Sie, dass die API einen Fehler zurückgibt, wenn Sie diese API über einen Standort aufrufen, der nur ein entfernter Standort in einer Flotte mit mehreren Standorten sein kann, da diese Art von Standort keinen Service-Endpunkt hat.

In der Tabelle der unterstützten Standorte finden Sie die Heimatregionen [Unterstützte AWS Standorte](#), in denen Flotten mit einem oder mehreren Standorten unterstützt werden.

Beispiel

```
aws gamelift list-locations --region ap-northeast-2
```

Dies AWS-Region unterstützt Flotten mit mehreren Standorten, weshalb mehrere Standorte zurückgegeben werden. Hier ist ein Beispiel für einen der Rückgabewerte:

```
[...]  
{  
  "LocationName": "ap-northeast-1",
```

```
"PingBeacon": {
  "UDPEndpoint": {
    "Domain": "gamelift-ping.ap-northeast-1.api.aws",
    "Port": 7770
  }
}
```

Important

Zwischenspeichern Sie die Ping-Beacon-Informationen, anstatt `ListLocations` sie vor jeder Latenzmessung aufzurufen. Die Domain- und Portinformationen sind statisch und die API ist nicht für Anfragen mit hohem Volumen konzipiert.

Implementierung von Latenzmessungen

Beachten Sie bei der Implementierung von Latenzmessungen mithilfe von UDP-Ping-Beacons die folgenden bewährten Methoden:

1. Speichern Sie Ping-Beacon-Informationen mit einem der folgenden Ansätze:
 - Kodieren Sie die Endpunkte in Ihrem Spielclient fest.
 - Zwischenspeichern Sie die Informationen in Ihrem Spiel-Backend.
 - Implementieren Sie einen Mechanismus zur regelmäßigen Aktualisierung (täglich/wöchentlich), um die Informationen zu aktualisieren.
2. Senden Sie UDP-Ping-Nachrichten:
 - Fügen Sie in den Nachrichtentext ein, was Sie wollen, solange er nicht leer ist und Sie Nachrichten unter der maximalen Größe von 300 Byte halten.
 - Beachten Sie für jeden Standort die folgenden Ratenbegrenzungen:
 - 3 Transaktionen pro Sekunde (TPS) pro eindeutiger Kombination aus IP-Adresse und Port des Absenders
 - 1000 TPS pro eindeutiger Absender-IP-Adresse
3. Latenz berechnen:
 - Senden Sie mehrere Pings an jeden Standort, um die durchschnittliche Latenz zu berechnen.
 - Erwägen Sie, gleichzeitige Pings an mehrere Standorte zu senden, um schnellere Ergebnisse zu erzielen.

- Verwenden Sie bei Bedarf die Wiederholungslogik, um neue Pakete für jedes Paket zu senden, das nicht innerhalb kurzer Zeit (normalerweise 1—3 Sekunden) zurückgegeben wurde, da bei UDP die Zustellung nicht zu 100% garantiert ist.
- Berechnet den Zeitunterschied zwischen dem Senden einer Nachricht und dem Empfangen der Antwort.
- Wenn ein großer Teil der UDP-Pings an einen Standort konsistent keine Antwort erhält, berechnen Sie die Latenz, indem Sie ICMP-Pings zu unseren [Amazon GameLift Servers Standarddienstendpunkten](#) als Fallback verwenden.

Tip

Wir empfehlen, dass Sie Port 7770 überall dort angeben, wo Sie die Liste der Ports dokumentieren, die Ihre Player in ihrem lokalen Netzwerk geöffnet haben müssen. Dies ist ein weiterer Grund, warum Sie einen Fallback zur Messung der Latenz (z. B. mit ICMP) haben sollten, falls dieser Port blockiert ist.

Codebeispiele

Hier sind einige einfache Beispiele, die zeigen, wie man UDP-Pings sendet und die Latenz berechnet.

C++

```
#include <iostream>
#include <cstring>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <chrono>

int main() {
    // Replace with Amazon GameLift Servers UDP ping beacon domain for your desired
    location
    const char* domain = "gamelift-ping.ap-south-1.api.aws";
    const int port = 7770;
    const char* message = "Ping";           // Your message
    const int num_pings = 3;                // Number of pings to send
```

```
// Create socket
int sock = socket(AF_INET, SOCK_DGRAM, 0);
if (sock < 0) {
    std::cerr << "Error creating socket" << std::endl;
    return 1;
}

// Resolve domain name to IP address
struct hostent* host = gethostbyname(domain);
if (host == nullptr) {
    std::cerr << "Error resolving hostname" << std::endl;
    close(sock);
    return 1;
}

// Set up the server address structure
struct sockaddr_in server_addr;
std::memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
std::memcpy(&server_addr.sin_addr, host->h_addr, host->h_length);

// Set socket timeout
struct timeval tv;
tv.tv_sec = 1; // 1 second timeout
tv.tv_usec = 0;
if (setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv)) < 0) {
    std::cerr << "Error setting socket timeout" << std::endl;
    close(sock);
    return 1;
}

double total_latency = 0;
int successful_pings = 0;

for (int i = 0; i < num_pings; ++i) {
    auto start = std::chrono::high_resolution_clock::now();

    // Send the message
    ssize_t bytes_sent = sendto(sock, message, std::strlen(message), 0,
                                (struct sockaddr*)&server_addr,
                                sizeof(server_addr));
```

```
    if (bytes_sent < 0) {
        std::cerr << "Error sending message" << std::endl;
        continue;
    }

    // Receive response
    char buffer[1024];
    socklen_t server_addr_len = sizeof(server_addr);
    ssize_t bytes_received = recvfrom(sock, buffer, sizeof(buffer), 0,
                                      (struct sockaddr*)&server_addr,
    &server_addr_len);

    auto end = std::chrono::high_resolution_clock::now();

    if (bytes_received < 0) {
        std::cerr << "Error receiving response or timeout" << std::endl;
    } else {
        std::chrono::duration<double, std::milli> latency = end - start;
        total_latency += latency.count();
        successful_pings++;

        std::cout << "Received response, latency: " << latency.count() << " ms"
    << std::endl;
    }

    // Wait a bit before next ping
    usleep(1000000); // 1s
}

// Close the socket
close(sock);

if (successful_pings > 0) {
    double avg_latency = total_latency / successful_pings;
    std::cout << "Average latency: " << avg_latency << " ms" << std::endl;
} else {
    std::cout << "No successful pings" << std::endl;
}

return 0;
}
```

C#

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Diagnostics;
using System.Threading.Tasks;

class UdpLatencyTest
{
    static async Task Main()
    {
        // Replace with Amazon GameLift Servers UDP ping beacon domain for your
        // desired location
        string domain = "gamelift-ping.ap-south-1.api.aws";
        int port = 7770;
        string message = "Ping";           // Your message
        int numPings = 3;                  // Number of pings to send
        int timeoutMs = 1000;              // Timeout in milliseconds

        await MeasureLatency(domain, port, message, numPings, timeoutMs);
    }

    static async Task MeasureLatency(string domain, int port, string message, int
numPings, int timeoutMs)
    {
        using (var udpClient = new UdpClient())
        {
            try
            {
                // Resolve domain name to IP address
                IPAddress[] addresses = await Dns.GetHostAddressesAsync(domain);
                if (addresses.Length == 0)
                {
                    Console.WriteLine("Could not resolve domain name.");
                    return;
                }

                IPEndPoint endPoint = new IPEndPoint(addresses[0], port);
                byte[] messageBytes = Encoding.UTF8.GetBytes(message);

                // Set receive timeout
                udpClient.Client.ReceiveTimeout = timeoutMs;
```



```
double totalLatency = 0;
int successfulPings = 0;
var stopwatch = new Stopwatch();

for (int i = 0; i < numPings; i++)
{
    try
    {
        stopwatch.Restart();

        // Send message
        await udpClient.SendAsync(messageBytes, messageBytes.Length,
endPoint);

        // Wait for response
        UdpReceiveResult result = await
ReceiveWithTimeoutAsync(udpClient, timeoutMs);

        stopwatch.Stop();
        double latency = stopwatch.Elapsed.TotalMilliseconds;

        totalLatency += latency;
        successfulPings++;

        string response = Encoding.UTF8.GetString(result.Buffer);
        Console.WriteLine($"Ping {i + 1}: {latency:F2}ms - Response:
{response}");
    }
    catch (SocketException ex)
    {
        Console.WriteLine($"Ping {i + 1}: Failed - {ex.Message}");
    }
    catch (TimeoutException)
    {
        Console.WriteLine($"Ping {i + 1}: Timeout");
    }

    // Wait before next ping
    await Task.Delay(1000); // 1s between pings
}

if (successfulPings > 0)
{
```

```

        double averageLatency = totalLatency / successfulPings;
        Console.WriteLine($"\\nSummary:");
        Console.WriteLine($"Successful pings: {successfulPings}/
{numPings}");
        Console.WriteLine($"Average latency: {averageLatency:F2}ms");
    }
    else
    {
        Console.WriteLine("\\nNo successful pings");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}

static async Task<UdpReceiveResult> ReceiveWithTimeoutAsync(UdpClient client,
int timeoutMs)
{
    using var cts = new System.Threading.CancellationTokenSource(timeoutMs);
    try
    {
        return await client.ReceiveAsync().WaitAsync(cts.Token);
    }
    catch (OperationCanceledException)
    {
        throw new TimeoutException("Receive operation timed out");
    }
}
}

```

Python

```

import socket
import time
import statistics
from datetime import datetime

def udp_ping(host, port, timeout=2):
    """
    Send a UDP ping and return the round trip time in milliseconds.
    """

```

```

Returns None if timeout occurs.
"""
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.settimeout(timeout)
message = b'ping'

try:
    # Resolve hostname first
    try:
        socket.gethostbyname(host)
    except socket.gaierror as e:
        print(f"Could not resolve hostname: {e}")
        return None

    start_time = time.time()
    sock.sendto(message, (host, port))

    # Wait for response
    data, server = sock.recvfrom(1024)
    end_time = time.time()

    # Calculate round trip time in milliseconds
    rtt = (end_time - start_time) * 1000
    return rtt

except socket.timeout:
    print(f"Request timed out")
    return None
except Exception as e:
    print(f"Error: {type(e).__name__}: {e}")
    return None
finally:
    sock.close()

def main():
    # Replace with Amazon GameLift Servers UDP ping beacon domain for your desired
    location
    host = "gamelift-ping.ap-south-1.api.aws"
    port = 7770
    num_pings = 3
    latencies = []

    print(f"\nPinging {host}:{port} {num_pings} times...")
    print(f"Start time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")

```

```
for i in range(num_pings):
    print(f"Ping {i+1}:")
    rtt = udp_ping(host, port)

    if rtt is not None:
        print(f"Response from {host}: time={rtt:.2f}ms")
        latencies.append(rtt)

    # Wait 1 second between pings
    if i < num_pings - 1:
        time.sleep(1)
    print()

# Calculate and display statistics
print("-" * 50)
print(f"Ping statistics for {host}:")
print(f"    Packets: Sent = {num_pings}, Received = {len(latencies)}, "
      f"    Lost = {num_pings - len(latencies)} "
      f"    f"({((num_pings - len(latencies)) / num_pings * 100):.1f}% loss)")

if latencies:
    print("\nRound-trip latency statistics:")
    print(f"    Minimum = {min(latencies):.2f}ms")
    print(f"    Maximum = {max(latencies):.2f}ms")
    print(f"    Average = {statistics.mean(latencies):.2f}ms")

print(f"\nEnd time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")

if __name__ == "__main__":
    main()
```

Die Amazon GameLift Servers-Versionshinweise bieten detaillierte Informationen über neue Funktionen, Updates und Fehlerbehebungen im Zusammenhang mit dem Service.

In den folgenden Tabellen sind alle Amazon GameLift Servers Versionen mit SDK-Versionsinformationen aufgeführt. Es ist nicht erforderlich, Comparable SDKs für deine Spielserver- und Client-Integrationen zu verwenden. Frühere Versionen eines SDK unterstützen jedoch möglicherweise nicht vollständig die neuesten Funktionen eines anderen.

Amazon GameLift Servers SDKs und Plugins sind Open Source. Siehe [Holen Sie sich Amazon GameLift Servers Entwicklungstools](#). Um die neuesten Versionen zu erhalten, wenden Sie sich an die [Amazon GameLift Servers GitHub Organisation](#).

1. **W**achstums-
 2. **E**ntwicklungs-
 3. **O**rganisations-
 4. **D**ienstleistungen
 5. **E**ntwicklungs-
 6. **O**rganisations-
 7. **E**ntwicklungs-
 8. **O**rganisations-
 9. **E**ntwicklungs-
 10. **O**rganisations-
 11. **E**ntwicklungs-
 12. **O**rganisations-
 13. **E**ntwicklungs-
 14. **O**rganisations-
 15. **E**ntwicklungs-
 16. **O**rganisations-
 17. **E**ntwicklungs-
 18. **O**rganisations-
 19. **E**ntwicklungs-
 20. **O**rganisations-
 21. **E**ntwicklungs-
 22. **O**rganisations-
 23. **E**ntwicklungs-
 24. **O**rganisations-
 25. **E**ntwicklungs-
 26. **O**rganisations-
 27. **E**ntwicklungs-
 28. **O**rganisations-
 29. **E**ntwicklungs-
 30. **O**rganisations-
 31. **E**ntwicklungs-
 32. **O**rganisations-
 33. **E**ntwicklungs-
 34. **O**rganisations-
 35. **E**ntwicklungs-
 36. **O**rganisations-
 37. **E**ntwicklungs-
 38. **O**rganisations-
 39. **E**ntwicklungs-
 40. **O**rganisations-
 41. **E**ntwicklungs-
 42. **O**rganisations-
 43. **E**ntwicklungs-
 44. **O**rganisations-
 45. **E**ntwicklungs-
 46. **O**rganisations-
 47. **E**ntwicklungs-
 48. **O**rganisations-
 49. **E**ntwicklungs-
 50. **O**rganisations-
 51. **E**ntwicklungs-
 52. **O**rganisations-
 53. **E**ntwicklungs-
 54. **O**rganisations-
 55. **E**ntwicklungs-
 56. **O**rganisations-
 57. **E**ntwicklungs-
 58. **O**rganisations-
 59. **E**ntwicklungs-
 60. **O**rganisations-
 61. **E**ntwicklungs-
 62. **O**rganisations-
 63. **E**ntwicklungs-
 64. **O**rganisations-
 65. **E**ntwicklungs-
 66. **O**rganisations-
 67. **E**ntwicklungs-
 68. **O**rganisations-
 69. **E**ntwicklungs-
 70. **O**rganisations-
 71. **E**ntwicklungs-
 72. **O**rganisations-
 73. **E**ntwicklungs-
 74. **O**rganisations-
 75. **E**ntwicklungs-
 76. **O**rganisations-
 77. **E**ntwicklungs-
 78. **O**rganisations-
 79. **E**ntwicklungs-
 80. **O**rganisations-
 81. **E**ntwicklungs-
 82. **O**rganisations-
 83. **E**ntwicklungs-
 84. **O**rganisations-
 85. **E**ntwicklungs-
 86. **O**rganisations-
 87. **E**ntwicklungs-
 88. **O**rganisations-
 89. **E**ntwicklungs-
 90. **O**rganisations-
 91. **E**ntwicklungs-
 92. **O**rganisations-
 93. **E**ntwicklungs-
 94. **O**rganisations-
 95. **E**ntwicklungs-
 96. **O**rganisations-
 97. **E**ntwicklungs-
 98. **O**rganisations-
 99. **E**ntwicklungs-
 100. **O**rganisations-
 101. **E**ntwicklungs-
 102. **O**rganisations-
 103. **E**ntwicklungs-
 104. **O**rganisations-
 105. **E**ntwicklungs-
 106. **O**rganisations-
 107. **E**ntwicklungs-
 108. **O**rganisations-
 109. **E**ntwicklungs-
 110. **O**rganisations-
 111. **E**ntwicklungs-
 112. **O**rganisations-
 113. **E**ntwicklungs-
 114. **O**rganisations-
 115. **E**ntwicklungs-
 116. **O**rganisations-
 117. **E**ntwicklungs-
 118. **O**rganisations-
 119. **E**ntwicklungs-
 120. **O**rganisations-
 121. **E**ntwicklungs-
 122. **O**rganisations-
 123. **E**ntwicklungs-
 124. **O**rganisations-
 125. **E**ntwicklungs-
 126. **O**rganisations-
 127. **E**ntwicklungs-
 128. **O**rganisations-
 129. **E**ntwicklungs-
 130. **O**rganisations-
 131. **E**ntwicklungs-
 132. **O**rganisations-
 133. **E**ntwicklungs-
 134. **O**rganisations-
 135. **E**ntwicklungs-
 136. **O**rganisations-
 137. **E**ntwicklungs-
 138. **O**rganisations-
 139. **E**ntwicklungs-
 140. **O**rganisations-
 141. **E**ntwicklungs-
 142. **O**rganisations-
 143. **E**ntwicklungs-
 144. **O**rganisations-
 145. **E**ntwicklungs-
 146. **O**rganisations-
 147. **E**ntwicklungs-
 148. **O**rganisations-
 149. **E**ntwicklungs-
 150. **O**rganisations-
 151. **E**ntwicklungs-
 152. **O**rganisations-
 153. **E**ntwicklungs-
 154. **O**rganisations-
 155. **E**ntwicklungs-
 156. **O**rganisations-
 157. **E**ntwicklungs-
 158. **O**rganisations-
 159. **E**ntwicklungs-
 160. **O**rganisations-
 161. **E**ntwicklungs-
 162. **O**rganisations-
 163. **E**ntwicklungs-
 164. **O**rganisations-
 165. **E**ntwicklungs-
 166. **O**rganisations-
 167. **E**ntwicklungs-
 168. **O**rganisations-
 169. **E**ntwicklungs-
 170. **O**rganisations-
 171. **E**ntwicklungs-
 172. **O**rganisations-
 173. **E**ntwicklungs-
 174. **O**rganisations-
 175. **E**ntwicklungs-
 176. **O**rganisations-
 177. **E**ntwicklungs-
 178. **O**rganisations-
 179. **E**ntwicklungs-
 180. **O**rganisations-
 181. **E**ntwicklungs-
 182. **O**rganisations-
 183. **E**ntwicklungs-
 184. **O**rganisations-
 185. **E**ntwicklungs-
 186. **O**rganisations-
 187. **E**ntwicklungs-
 188. **O**rganisations-
 189. **E**ntwicklungs-
 190. **O**rganisations-
 191. **E**ntwicklungs-
 192. **O**rganisations-
 193. **E**ntwicklungs-
 194. **O**rganisations-
 195. **E**ntwicklungs-
 196. **O**rganisations-
 197. **E**ntwicklungs-
 198. **O**rganisations-
 199. **E**ntwicklungs-
 200. **O**rganisations-
 201. **E**ntwicklungs-
 202. **O**rganisations-
 203. **E**ntwicklungs-
 204. **O**rganisations-
 205. **E**ntwicklungs-
 206. **O**rganisations-
 207. **E**ntwicklungs-
 208. **O**rganisations-
 209. **E**ntwicklungs-
 210. **O**rganisations-
 211. **E**ntwicklungs-
 212. **O**rganisations-
 213. **E**ntwicklungs-
 214. **O**rganisations-
 215. **E**ntwicklungs-
 216. **O**

der
päter

Frühere Versionen

Verfügbare
Ebenen-
code
Dienstes

Unwirklic
Unity
H-C
Unwirklic
n+)

[1052](#)
[der](#)
[später](#)

[1052](#)
[der](#)
[später](#)

[1050](#)
[der](#)
[später](#)

[1052](#)
[der](#)
[später](#)

[1052](#)
[der](#)
[später](#)

[1052](#)
[der](#)
[später](#)

Verfügbare meK

Ben-

Ben-

Dienstes

Benwirklic

hity

HC

Benwirklic

h+)

[052](#)

[NET](#)

[päter](#)

[052](#)

[der](#)

[päter](#)

[051](#)

[der](#)

[päter](#)

[051](#)

[der](#)

[päter](#)

[070](#)

[der](#)

[päter](#)

[051](#)

[der](#)

[päter](#)

[051](#)

[der](#)

[päter](#)

Verfügbare	meK
Ben-	
Ben-	
Dienstes	
Benwirklic	
hity	
H(C	
Benwirklic	
n+)	
2112051	
der	
päter	
2112051	
der	
päter	
2112052	
der	
päter	
2112052	
der	
päter	
2112052	
der	
päter	
2112052	
der	
päter	
2112052	
der	
päter	

Verfügbare

Regionen

Amazon

Dienste

Unwirklic

hity

HC

Unwirklic

h+)

[5.10.202](#)

[der](#)

[päter](#)

[5.10.202](#)

[der](#)

[päter](#)

[5.10.202](#)

[der](#)

[päter](#)

[5.10.202](#)

[der](#)

[päter](#)

[5.10.202](#)

[der](#)

[päter](#)

[5.10.202](#)

[der](#)

[päter](#)

[5.10.202-0](#)

[der](#)

[päter](#)

Verfügt über
Einen
einen
Dienstes

Einwirklic
hity
H(C
Einwirklic
n+)

[50200-3](#)
[der](#)
[päter](#)

[50200-02](#)
[NET](#)
[NET](#)
(6)

[50200-02](#)
[der](#)
[päter](#)

[50200-02](#)
[der](#)
[päter](#)

[50200-02](#)
[der](#)
[päter](#)

[50200-02](#)
[der](#)
[päter](#)

Verfügbare

Regionen

ausgewählter

Dienstes

Unwirklic

hity

HC

Unwirklic

h+)

[10.13.2002](#)

der

später

[10.13.2002](#)

der

später

[10.13.2002](#)

der

später

[10.13.2002](#)

der

später

[10.13.2002](#)

der

später

[10.13.2002](#)

der

später

[12.10.2013](#)

der

höher

[17.0.0-0](#)

der

höher

Dienstes

[17.0.0-0](#)

der

höher

Dienstes

höher

[17.0.0-0](#)

der

höher

[17.0.0-0](#)

der

höher

[17.0.0-0](#)

der

höher

[17.0.0-0](#)

der

höher

[17.0.0-0](#)

der

höher

[17.0.0-0](#)

der

höher

[17.0.0-0](#)

der

höher

[13.0.0-1](#) meK

[13.0.0-1](#) den-

[13.0.0-1](#) al

Dienstes

[13.0.0-1](#) nwirklic

[13.0.0-1](#) hity

[13.0.0-1](#) (C

[13.0.0-1](#) nwirklic

[13.0.0-1](#) n+)

[13.0.0-2](#)

[13.0.0-2](#) der

[13.0.0-2](#) öher

[13.0.0-201](#)

[13.0.0-201](#) der

[13.0.0-201](#) öher

[13.0.0-201](#)

[13.0.0-201](#) der

[13.0.0-201](#) öher

[13.0.0-12-1](#)

[13.0.0-12-1](#) der

[13.0.0-12-1](#) öher

[12.0.0-201](#)

[12.0.0-201](#) der

[12.0.0-201](#) öher

[12.0.0-201](#)

[12.0.0-201](#) der

[12.0.0-201](#) öher

[12.0.0-201](#)

[12.0.0-201](#) der

[12.0.0-201](#) öher

[1.1.102-1](#)

der

höher

höher

[1.1.102-0](#)

der

höher

höher

höher

[1.1.102-1](#)

der

höher

höher

[1.1.102-0](#)

der

höher

höher

[1.1.102-1](#)

der

höher

höher

[1.1.108-1](#)

der

höher

höher

[1.1.102-1](#)

der

höher

höher

[1.1.102-1](#)

der

höher

höher

[1.1.102-2](#)

der

höher

Verfügbare

Erstgen-

erstellte

Dienstes

Unwirklic

Unity

HC

Unwirklic

er+)

[3.0.13.201](#)

der

höher

[3.0.10.201](#)

der

höher

[3.0.9.0](#)

der

höher

[3.0.7.0](#)

der

höher

Versionshinweise

Die folgenden Versionshinweise sind in chronologischer Reihenfolge, wobei die neuesten Updates zuerst aufgeführt sind. Amazon GameLift Servers wurde erstmals im Jahr 2016 veröffentlicht.

Versionshinweise mit einem Datum vor den hier aufgeführten finden Sie unter den Links zum Veröffentlichungsdatum in [SDK-Versionen](#).

4. März 2026: bietet Amazon GameLift Servers jetzt DDoS-Schutz für Linux-basierte EC2- und Container-Flotten auf SDKv5

Amazon GameLift Servers bietet jetzt distributed-denial-of-service (DDoS-) Schutz für Linux-basierte EC2- und Container-Flotten auf SDKv5. Das Player-Gateway-Proxy-Relay-Netzwerk befindet sich zusammen mit Ihren Spieleserver-Instanzen und bietet eine Überprüfung des Datenverkehrs, eine Geschwindigkeitsbegrenzung pro Spieler und die Verschleierung der IP-Adressen von Spieleservern — alles mit vernachlässigbarer zusätzlicher Latenz und ohne zusätzliche Kosten. Diese Funktion ist zunächst in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Europa (Frankfurt), Europa (Irland), Asien-Pazifik (Sydney), Asien-Pazifik (Tokio), Asien-Pazifik (Seoul). Weitere Informationen zu Servicestandorten finden Sie unter [Amazon GameLift Servers Servicestandorte](#).

Weitere Informationen:

- [Blogbeitrag: Wir stellen vor: Amazon GameLift Servers DDoS-Schutz](#)
- [DDoS-Schutz mit Amazon GameLift Servers](#), Amazon GameLift Servers Entwicklerhandbuch
- [So funktioniert das Player-Gateway](#), Amazon GameLift Servers Entwicklerhandbuch
- [Integrieren Sie das Player-Gateway in ein Spiel](#), Amazon GameLift Servers Entwicklerhandbuch
- [Client-SDK SDK for C++](#)
- [Client-SDK für Unreal](#). Für den Zugriff ist eine Mitgliedschaft in der Epic Games-Organisation am GitHub erforderlich. Einzelheiten [findest du unter Unreal Engine GitHub auf](#).

24. Februar 2026: Amazon GameLift Servers Plugins unterstützen jetzt Unity 6.3 und Unreal Engine 5.7

Aktualisierte Plugin-Versionen:

- [Plugin für Unity](#), Version 3.2.1 (beinhaltet das Server-SDK SDK for Unity, Version 5.4.0)
- [Plugin für Unreal](#), Version 3.1.1 (enthält das Server-SDK für Unreal, Version 5.4.0)

Amazon GameLift Servers veröffentlicht neue Versionen des Plugins und des Server-SDK für Unreal Engine und Unity. Diese neuesten Versionen beinhalten die folgenden Updates:

- Sie unterstützen jetzt die Verwendung mit Unreal Engine 5.7 und Unity 6.3.

Weitere Informationen:

- [Integration von Spielen mit dem Amazon GameLift Servers Plugin für Unreal Engine, Entwicklerhandbuch Amazon GameLift Servers](#)
- [Integration Amazon GameLift Servers in ein Unreal Engine-Projekt, Entwicklerhandbuch Amazon GameLift Servers](#)
- [Integration von Spielen mit dem Amazon GameLift Servers Plugin für Unity, Amazon GameLift ServersEntwicklerhandbuch](#)
- [Integration Amazon GameLift Servers in ein Unity Engine-Projekt, Amazon GameLift ServersEntwicklerhandbuch](#)

29. Januar 2026: unterstützt Amazon GameLift Servers jetzt die Skalierung von Fleets ohne Instanzen to/from

Amazon GameLift Serversunterstützt jetzt die automatische Skalierung auf und von null Instanzen basierend auf der Aktivität der Spielsitzung. Flotten werden nach einem bestimmten Zeitraum ohne Spielsitzungen auf Null herunterskaliert und von Null hochskaliert, wenn Spielsitzungen angefordert werden, was eine Option zur Kostenoptimierung bietet.

Amazon GameLift ServersAPI-Referenzlinks:

- [AWS SDK-Aktion UpdateFleetCapacity](#)

Weitere Informationen:

- [Verwaltung der Skalierung einer Amazon GameLift Server-Flotte To/From Zero](#)

22. Januar 2026: Amazon GameLift Servers Realtime unterstützt jetzt Node.js 24.x auf Amazon Linux 2023

Amazon GameLift ServersRealtimeunterstützt jetzt Node.js 24.x auf Amazon Linux 2023. Dieses Update bietet Zugriff auf die neuesten Funktionen und Sicherheitsverbesserungen von Node.js, während es auf dem modernen Betriebssystem Amazon Linux 2023 ausgeführt wird.

Sie können jetzt bei der Erstellung einer Realtime Flotte auch ein optionales Installationsskript hinzufügen, das die Installation von Software während des Instance-Starts für Ihre spezifischen

Anforderungen ermöglicht. Um die neue Runtime-Version zu verwenden, geben Sie beim Erstellen von Realtime Skripts Node.js 24.x an.

Amazon GameLift ServersAPI-Referenzlinks:

- [AWS SDK-Aktion CreateScript](#)

Weitere Informationen:

- [Laufzeitversionen von Node.js](#)
- [Fügen Sie ein Installationsskript hinzu](#)

9. Dezember 2025: bietet Amazon GameLift Servers jetzt KI-gestützte Unterstützung in der Webkonsole

Amazon GameLift Servers bietet jetzt KI-gestützte Unterstützung in der Webkonsole und nutzt Amazon Q Developer, um Spieleentwicklern maßgeschneiderte Anleitungen zu bieten. Diese neue Funktion integriert Amazon GameLift Servers Fachwissen, um Kunden dabei zu helfen, sich in komplexen Workflows zurechtzufinden, Probleme zu beheben und ihre Spieleserverbereitungen effizienter zu optimieren.

Diese neue Funktion bietet KI-gestützte Empfehlungen für die Integration von Spieleservern, die Flottenkonfiguration und die Leistungsoptimierung direkt in der Amazon GameLift Servers Konsole. Diese Verbesserung zielt darauf ab, die Entscheidungsprozesse zu rationalisieren, die Zeit für die Fehlerbehebung zu verkürzen und die allgemeine Nutzung der Ressourcen zu verbessern, was zu Kosteneinsparungen und besseren Spielerlebnissen führt.

Weitere Informationen:

- [Amazon GameLift Servers-Konsole](#)

6. November 2025: Unterstützt Amazon GameLift Servers jetzt Spiele-Builds, die das Betriebssystem Windows Server 2022 verwenden

Amazon GameLift Servers Kunden können jetzt das Betriebssystem Windows Server 2022 verwenden, um ihre Spieleserver zu hosten. Windows Server 2022 bietet mehrere Verbesserungen gegenüber Windows Server 2016, einschließlich der Sicherheit. Dieses Betriebssystem ist in allen AWS-Regionen mit Ausnahme der Regionen China verfügbar.

Kunden können die neueren Windows-Betriebssysteme verwenden und weiterhin wichtige Sicherheitsupdates erhalten, wenn der Support für Windows Server 2016 im Januar 2027 endet. Die Support für Windows Server 2022 läuft bis zum 13. Oktober 2031.

Amazon GameLift ServersAPI-Referenzlinks:

- [AWS SDK-Aktion CreateBuild](#)
- [CLI-Befehl upload-build](#)
- [CLI-Befehl create-build](#)

28. Oktober 2025: Amazon GameLift Servers Fügt allen Servern SDKs und Plugins den Metrik-Collector OpenTelemetry (OTel) hinzu

Amazon GameLift Servers beinhaltet jetzt die integrierte Erfassung von OpenTelemetry (OTel) Metriken auf allen Servern SDKs und Plugins und bietet so eine standardisierte Beobachtbarkeit für die Leistungsüberwachung von Spieleservern. Diese Integration bringt branchenübliche Telemetriefunktionen direkt auf Ihre Spieleserver und ermöglicht so eine umfassende Überwachung und Fehlerbehebung, ohne dass zusätzliche Instrumente oder Konfigurationen erforderlich sind.

Mit der OpenTelemetry Integration können Sie jetzt:

- Erfassen Sie mithilfe standardisierter Protokolle automatisch wichtige Metriken, Logs und Traces des Spieleservers
- Integrieren Sie sich nahtlos in bestehende Überwachungsplattformen und Observability-Tools
- Überwachen Sie die Leistung und den Zustand der Spieleserver in Echtzeit auf allen Flotteninstanzen
- Beheben Sie Probleme schneller mit Telemetriedaten, die in verschiedenen Überwachungslösungen funktionieren

OpenTelemetry ist ein Open-Source-Observability-Framework, das eine standardisierte Methode zum Sammeln, Verarbeiten und Exportieren von Telemetriedaten bietet.

Weitere Informationen:

- [Überwachung mit Server-Telemetrie-Metriken](#), Entwicklerhandbuch Amazon GameLift Servers

2. Oktober 2025: Amazon GameLift Servers Fügt konsolenbasierten Fernzugriff auf Flotteninstanzen hinzu

Amazon GameLift Servers bietet jetzt direkten Fernzugriff auf verwaltete EC2- und Container-Flotteninstanzen über die Amazon GameLift Servers Konsole, sodass kein Authentifizierungsmanagement erforderlich ist. Diese neue Funktion verwendet Amazon EC2 Systems Manager (SSM), um einen sicheren, browserbasierten Terminalzugriff zu ermöglichen, ohne dass eine zusätzliche Einrichtung oder Verwaltung von Anmeldeinformationen erforderlich ist.

Mit dem konsolenbasierten Fernzugriff können Sie jetzt:

- Probleme mit der Integration von Spieleservern in Echtzeit beheben
- Überwachen Sie die Aktivität des Spieleservers und greifen Sie während aktiver Spielsitzungen auf Protokolldateien zu
- Führen Sie Diagnose- und Benchmarking-Tools anhand des tatsächlichen Spielerverkehrs aus

Diese Funktion ist für Flotten verfügbar, auf denen das Server-SDK Version 5.x ausgeführt wird. Für Flotten, auf denen frühere SDK-Versionen ausgeführt werden, können Sie weiterhin die bestehende AWS CLI-Methode für den Fernzugriff verwenden.

Um eine Remoteverbindung zu einer Flotteninstanz herzustellen, öffnen Sie die Amazon GameLift Server-Konsole und suchen Sie die verwaltete EC2- oder Containerflotte, auf die Sie zugreifen möchten. Öffnen Sie in den Flottendetails den Tab Instances, wählen Sie eine Instance aus der Liste aus und wählen Sie Connect aus.

Weitere Informationen:

- [Stellen Sie eine Remoteverbindung zu Amazon GameLift Servers Flotteninstanzen her](#), Amazon GameLift Servers Entwicklerhandbuch

23. September 2025: Amazon GameLift Servers Fügt Unterstützung für die neue lokale Zone in Dallas, Texas, hinzu

Mit Amazon GameLift Servers verwaltetem Hosting können Sie jetzt Spielserverressourcen in einer neuen lokalen Zone in Dallas, TX, bereitstellen (us-east-1-dfw-2). Local Zones bieten dir die Möglichkeit, deine Spieleserver geografisch näher an deinen Spielern zu platzieren, um die Latenz zu reduzieren und das Gameplay deutlich zu verbessern. In dieser neuen lokalen Zone können Sie

die folgenden EC2-Instance-Typen verwenden: C6gn, C6i, C6in, M6g, M6i, M6in, M8g und R6i. Detaillierte Beschreibungen [AWS der einzelnen Instance-Typen finden Sie unter EC2-Instance-Typen](#).

Um mit der Ausrichtung von Spielsitzungen in der neuen Dallas Local Zone zu beginnen, melden Sie sich zunächst für die lokale Zone für Sie AWS-Konto an. Anschließend können Sie es als Remote-Standort zu einer neuen oder bestehenden Flotte mit mehreren Standorten hinzufügen. Falls dein Spiel dies nutzt Amazon GameLift ServersFlexMatch, aktualisiere alle Flotten in deiner Matchmaking-Warteschlange, sodass sie auch die neue lokale Zone enthalten. Bei Flotten mit mehreren Standorten kannst du die Hosting-Kapazitäten an jedem Standort direkt verwalten.

Die Muttergesellschaft AWS-Region der neuen Local Zone in Dallas, TX, ist us-east-1 (Virginia). Die ursprüngliche lokale Zone in Dallas, TX (us-east-1-dfw-1) akzeptiert keine neuen Opt-in-Anfragen mehr.

Weitere Informationen:

- [Erste Schritte mit AWS Local Zones](#), AWS Local Zones Benutzerhandbuch
- [Amazon GameLift ServersServicestandorte](#), Amazon GameLift ServersEntwicklerhandbuch
- [Flottenstandorte aktualisieren](#), Amazon GameLift ServersEntwicklerhandbuch

12. August 2025: Amazon GameLift Servers Das Plugin für Unreal Engine unterstützt jetzt die UE-Version 5.6

Aktualisierte Plugin-Version:

- [Plugin für Unreal](#), Version 3.0.1 (enthält das Server-SDK für Unreal, Version 5.3.1)

Amazon GameLift Serversveröffentlicht neue Versionen des Plugins für Unreal Engine und des Server-SDK für Unreal Engine. Diese neuesten Versionen beinhalten die folgenden Updates:

- Sie unterstützen jetzt die Verwendung mit Unreal Engine 5.6.
- Kompilierungsprobleme beim Packen von Android-Client-Zielen wurden behoben.

Weitere Informationen:

- [Integration von Spielen mit dem Amazon GameLift Servers Plugin für Unreal Engine](#), Amazon GameLift ServersEntwicklerhandbuch

- [Integration Amazon GameLift Servers in ein Unreal Engine-Projekt, Entwicklerhandbuch Amazon GameLift Servers](#)

24. Juni 2025: Amazon GameLift Servers Startet UDP-Ping-Beacons zur Messung der Netzwerklatenz in Echtzeit für Multiplayer-Spiele

Aktualisierte SDK-Versionen:

- AWS-SDK 1.11.595

Amazon GameLift Servers veröffentlicht zur allgemeinen Verfügbarkeit eine Reihe fester Endpunkte, sogenannte UDP-Ping-Beacons, mit denen Sie die Latenz zwischen Spielergeräten und Spielerverstandorten genau messen können. Die UDP-Ping-Beacon-Endpunkte sind in allen AWS globalen Regionen und Local Zones verfügbar, die von unterstützt werden Amazon GameLift Servers, mit Ausnahme der Regionen AWS China.

Die meisten Multiplayer-Spiele verwenden UDP (User Datagram Protocol) als primäres Paketübertragungsprotokoll, da es Leistungsvorteile für Echtzeitspiele bietet. Das Verständnis und die Optimierung der Netzwerklatenz sind entscheidend, um das bestmögliche Spielerlebnis zu bieten. UDP-Ping-Beacons bieten eine konsistente und zuverlässige Methode, um die tatsächliche UDP-Paketlatenz zwischen Spielern und Spieleservern zu messen, sodass Sie bessere Entscheidungen über das player-to-server Matching und die Platzierung von Spielsitzungen treffen können.

Die [ListLocations](#) API wurde um Informationen zur Endpunktdomäne und zum Port als Teil der Liste der zurückgegebenen Standorte erweitert, sodass der programmgesteuerte Zugriff auf die Endgeräte vereinfacht wird.

Ihr Spielclient kann UDP-Nachrichten an diese Endpunkte senden und asynchrone Antworten mit denselben Daten empfangen. So erhalten Sie Latenzmessungen, die die tatsächlichen Verkehrsbedingungen im Spiel zwischen dem Gerät eines Spielers und potenziellen Hosting-Standorten besser wiedergeben. Diese Endpunkte sind permanent und bleiben verfügbar, solange das Hosten von Spielen an Amazon GameLift Servers diesem Standort unterstützt wird.

Weitere Informationen:

- [UDP-Ping-Beacons](#), Amazon GameLift Servers Entwicklerhandbuch
- [ListLocations](#), Amazon GameLift Servers API-Referenz

24. Juni 2025: Amazon GameLift Servers Fügt Standortunterstützung für Thailand und Malaysia hinzu

Mit Amazon GameLift Servers verwaltetem Hosting können Sie jetzt Spieleserverressourcen in Bangkok, Thailand, und Kuala Lumpur, Malaysia, bereitstellen und so die Reichweite Ihrer Spiele auf Spieler in ganz Südostasien ausdehnen. Diese neuen Regionen tragen dazu bei, die Latenz zu reduzieren und das Spielerlebnis für Spieler in diesen Gebieten zu verbessern.

Die folgenden Standorte AWS-Regionen sind als abgelegene Standorte für Flotten mit mehreren Standorten verfügbar. Um mit der Ausrichtung von Spielsitzungen an diesen Orten zu beginnen, fügen Sie sie als abgelegene Standorte zu einer neuen oder bestehenden Flotte mit mehreren Standorten hinzu. Bei Flotten mit mehreren Standorten kannst du die Hosting-Kapazitäten an jedem Standort direkt verwalten.

- Asien-Pazifik (Thailand) (`ap-southeast-7`)
- Asien-Pazifik (Malaysia) (`ap-southeast-5`)

Diese AWS-Regionen sind standardmäßig nicht für ein AWS Konto aktiviert. Sie müssen sich für jede Region anmelden, bevor Sie dort Amazon GameLift Servers Ressourcen bereitstellen können.

Weitere Informationen:

- [Amazon GameLift Servers Servicestandorte](#), Leitfaden Amazon GameLift Servers für Hosting-Entwickler
- [Flottenstandorte aktualisieren](#), Leitfaden Amazon GameLift Servers für Hosting-Entwickler
- [Aktivieren oder deaktivieren Sie AWS Regionen in Ihrem AWS Konto](#), Referenzhandbuch zur Kontoverwaltung.

29. Mai 2025: Amazon GameLift Servers veröffentlicht neue Versionen des Server-SDK und der Plugins für Unreal Engine und Unity

Aktualisierte SDK-Versionen:

- [C++-Server-SDK](#), Version 5.3.0
- [C#-Server-SDK](#), Version 5.3.0
- [Go-Server-SDK](#), Version 5.3.0

- [Server-SDK für Unreal, Version 5.3.0](#)
- [Server-SDK SDK for Unity](#), Version 5.3.0

Aktualisierte Plugin-Versionen:

- [Plugin für Unreal](#), Version 3.0.0
- [Plugin für Unity, Version 3.1.0](#)

Neue Versionen des Server-SDK SDK for C++, C#, Go, Unreal und Unity sowie neue Plugin-Versionen für Unreal Engine und Unity sind jetzt Open Source. [Sie sind alle in der Organisation verfügbar. Amazon GameLift Servers GitHub](#) Detaillierte Informationen zum Update finden Sie in den Versionshinweisen und Readmes in den einzelnen Repositories.

Wichtige Server-SDK-Updates:

- Verbesserte clientseitige Validierung und Fehlerantworten auf allen Servern. SDKs
- Die [OnProcessTerminate](#) Callback-Funktion hat jetzt eine Standardlogik, um den Spieleserverprozess zu beenden.
- Die Funktion verwendet [InitSDK\(\)](#) jetzt ein Idempotenz-Token, um mehrere Wiederholungen zu unterstützen.
- Der [OnUpdateGameSession](#) Callback kann jetzt Daten für übergeben. `autoBackfillMode`

Wichtige Plugin-Updates:

- Das Plugin für Unreal Engine verfügt jetzt über einen optimierten Installations- und Einrichtungsprozess mit mehr Automatisierung und weniger Voraussetzungen (CMake, OpenSSL und die Unreal Cross-Compile Toolchain).
- Das Plugin für Unreal Engine hat die Benutzeroberfläche für den Managed EC2-Workflow verbessert, einschließlich der Unterstützung von Leerzeichen in den Buildpfaden des Spielclients und des Servers. Außerdem kannst du jetzt Befehlszeilenargumente hinzufügen, wenn du einen Spielclient vom Editor aus startest.
- Das Plugin für Unreal Engine unterstützt jetzt integrierte ARM-Server-Builds. UE5

Weitere Informationen:

- [Amazon GameLift ServersPlugin für Unreal Engine](#), Amazon GameLift ServersEntwicklerhandbuch

- [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#), Amazon GameLift ServersEntwicklerhandbuch
- [Server-SDK 5.x für Amazon GameLift Servers](#), Amazon GameLift ServersEntwicklerhandbuch
- [Amazon GameLift ServersIn ein Unreal Engine-Projekt integrieren](#), Amazon GameLift ServersEntwicklerhandbuch
- [Amazon GameLift ServersIn ein Unity-Projekt integrieren](#), Amazon GameLift ServersEntwicklerhandbuch

24. April 2025: Das C#-Server-SDK 5 unterstützt Amazon GameLift Servers vorerst .NET 8

Aktualisierte SDK-Versionen:

- C# Server-SDK, Version 5.2.1

Spieleentwickler, die C# verwenden, können jetzt .NET 8 als Zielframework für ihre Projekte verwenden. Amazon GameLift Servers Mit.NET 8 können Sie von Leistungsverbesserungen profitieren, einschließlich verbesserter just-in-time (JIT-) Kompilierung, Optimierung der Speichernutzung und schnellerer Startzeiten. Wenn Sie derzeit .NET 6 verwenden, empfehlen wir Ihnen, eine Migration zu .NET 8 zu planen, einschließlich der Aktualisierung Ihres C#-Server-SDK auf die neueste Version. Microsoft kündigte die Unterstützung von.NET 8 mit Sicherheitspatches und technischen Updates bis November 2026 an.

[Laden Sie die neueste Version des C#-Server-SDK herunter für Amazon GameLift Servers](#)

Weitere Informationen:

- [Holen Sie sich Amazon GameLift Servers Entwicklungstools](#), Amazon GameLift ServersEntwicklerhandbuch

27. März 2025: Amazon GameLift Servers erweitert die Unterstützung für EC2-Instance-Familien der nächsten Generation

Aktualisierte SDK-Versionen:

- AWS SDK 1.11.535

Sie können Ihr Gameserver-Hosting jetzt optimieren, indem Sie aus einer größeren Auswahl an Amazon EC2 EC2-Instances der Instance-Familien der 5. bis 8. Generation auswählen. Amazon GameLift Servers Jede neue EC2-Generation bietet Fortschritte in den Bereichen EC2-Rechenleistung, Arbeitsspeicher und Netzwerk. Die Instances der 8. Generation bieten hochmoderne AWS Graviton4- und Intel Xeon-basierte Instances. Instances der nächsten Generation sind in den folgenden Instance-Familien verfügbar:

- [Universell einsetzbar](#) (M-Serie)
- Für die [Datenverarbeitung optimiert](#) (C-Serie)
- [Speicheroptimiert](#) (R-Serie)

Sie können auch Varianten wählen, die lokalen Speicher (d), erweiterte Netzwerkfunktionen (n) und spezifische Prozessorarchitekturen (Intel/AMD/Graviton – i/a/g) bieten. Die Instances der nächsten Generation sind in allen Regionen verfügbar, AWS-Regionen die von unterstützt werdenAmazon GameLift Servers, mit Ausnahme der Regionen AWS China. Weitere Details finden Sie unter [Amazon GameLift ServersServicestandorte](#).

Verwenden Sie diese neuen Instance-Typen mit Amazon GameLift Servers verwalteten EC2-Flotten und verwalteten Container-Flotten. Wenn Sie Ihr bestehendes Game-Hosting auf einen neuen Instance-Typ (dieselbe Architektur) umstellen, stellen Sie einfach neue Flotten bereit, wobei alle Konfigurationseinstellungen mit Ausnahme des Instance-Typs unverändert bleiben.

Weitere Informationen:

- [Amazon GameLift ServersPreise für Instanzen](#)
- [CreateFleetAmazon GameLift Servers](#)API-Referenz
- [CreateContainerFleet](#), Amazon GameLift ServersAPI-Referenz

11. März 2025: Amazon GameLift Servers bringt neuen Game Server Wrapper für schnelles Onboarding auf den Markt

Der neue Gameserver-Wrapper für reduziert die Zeit, die benötigt wird, um einen Spielservers zu hosten, Amazon GameLift Servers erheblich. Amazon GameLift Servers Da keine Codeänderungen erforderlich sind, können Sie den Wrapper verwenden, um Ihrem Spiel grundlegende Funktionen zur Verwaltung von Spielsitzungen hinzuzufügen und es für eine Amazon GameLift Servers Anywhere-Flotte, eine verwaltete EC2-Flotte oder eine verwaltete Containerflotte bereitzustellen. Dieses Tool

ist ideal für die praktische Evaluierung von Amazon GameLift Servers Funktionen auf Ihrem eigenen Spieleserver oder einem aus einem Beispielspiel. Es ist auch nützlich, um Spieleserver-Iterationen schnell bereitzustellen, z. B. für Rapid Prototyping oder Tests.

Mit den grundlegenden Funktionen zur Verwaltung von Spielsitzungen kann Ihr Spieleserver eine Verbindung mit dem Amazon GameLift Servers Dienst initialisieren, auf Aufforderungen zum Starten und Beenden von Spielsitzungen reagieren und nach Abschluss einer Spielsitzung herunterfahren.

[Laden Sie den Gameserver-Wrapper von herunter. GitHub](#) In der Readme-Datei erfahren Sie, wie Sie den Wrapper installieren und mit Ihren Spielprojekten verwenden können. GitHub

6. März 2025: Amazon GameLift ist jetzt Amazon GameLift Servers und Amazon GameLift Streams

Amazon GameLift ist ein vollständig verwalteter Service, der Entwickler dabei unterstützt, die anspruchsvollsten Spiele der Welt zu entwickeln, zu skalieren und bereitzustellen. Mit der allgemein verfügbaren Version von Amazon GameLift Streams bietet Amazon GameLift nun sowohl hochskalierbare Gameserver als auch Streaming-Funktionen für flüssiges Gameplay.

[Amazon GameLift Servers](#) bietet Spieleentwicklern die Möglichkeit, dedizierte Spieleserver bereitzustellen, zu betreiben und zu skalieren. Sie können innerhalb weniger Minuten leistungsstarke Spieleserver in der Cloud bereitstellen und nach oben oder unten skalieren, um der Nachfrage der Spieler gerecht zu werden. Es basiert auf einer AWS bewährten Computerumgebung und Amazon GameLift Servers unterstützt 100 Millionen gleichzeitige Spieler in einem einzigen Spiel, 100.000 Spielerzugänge pro Sekunde und 9.000 neue Recheninstanzen pro Minute. Und mit Sicherheit auf Unternehmensniveau, Matchmaking für die meisten Zuschauer und pay-as-you-go Flexibilität erleichtert es Ihnen den Einstieg, egal ob Sie an einer neuen Spielidee arbeiten oder ein Spiel mit Millionen von Spielern ausführen.

[Amazon GameLift Streams](#) unterstützt Spieleentwickler dabei, Game-Streaming-Erlebnisse mit einer Auflösung von bis zu 1080p und frames-per-second 60 Bildern pro Sekunde ohne wahrnehmbare Verzögerung auf allen Geräten wie iOS, Android, FireOS und PCs für Gamer bereitzustellen. Mithilfe eines einzigen AWS Angebots können Publisher ihre Spielinhalte innerhalb von Minuten ohne Änderungen auf vollständig verwalteten Cloud-basierten GPU-Instances bereitstellen und sie über das AWS Network Backbone direkt auf jedes Endspieler-Gerät mit einem Webbrowser bereitstellen. Spieler können in nur wenigen Sekunden mit dem Spielen beginnen, ohne auf einen Download oder eine Installation warten zu müssen, und es bietet ein Spielerlebnis, das kaum von einem lokalen Spiel auf einem PC oder einer Spielekonsole zu unterscheiden ist.

14. Januar 2025: Amazon GameLift Servers veröffentlicht erweiterte Unterstützung für die Priorisierung der Platzierung von Spielsitzungen

Aktualisierte SDK-Versionen:

- AWS SDK 1.11.485

Als Reaktion auf Kundenfeedback veröffentlichen wir eine neue Funktion, mit der ihr Standorte für Anfragen zur Platzierung einzelner Spielsitzungen priorisieren könnt. Für deine Warteschlangen, die so konfiguriert sind, dass die Platzierung nach Standort priorisiert wird, kannst du jetzt bei jeder Platzierungsanfrage eine benutzerdefinierte Liste mit bevorzugten Standorten bereitstellen.

Mit dieser neuen Funktion können Kunden die Standortprioritäten für jede Platzierungsanfrage nach Bedarf dynamisch ändern. Durch die zusätzliche Flexibilität können Sie besser auf sich ändernde Bedingungen wie Spielerstandorte, Flottenauslastung oder Serverstatus reagieren. Es kann auch Kunden unterstützen, die die Auswahl der Einsatzorte weiter anpassen möchten.

Weitere Informationen:

- [Priorisieren Sie die Platzierung von Spielsitzungen](#), Amazon GameLift ServersEntwicklerhandbuch
- [StartGameSessionPlacement](#), Amazon GameLift ServersAPI-Referenz

2. Januar 2025: Amazon GameLift Servers veröffentlicht neue Unterstützung für das Beenden von Spielsitzungen

Aktualisierte SDK-Versionen:

- AWS SDK 1.11.477

Als Reaktion auf Kundenfeedback veröffentlichen wir neue Funktionen, mit denen du einzelne Spielsitzungen einfacher beenden kannst. Mit dieser Version kannst du eine Spielsitzung jetzt direkt in der Amazon GameLift Servers Konsole oder mithilfe des AWS CLI oder AWS SDK für beendenAmazon GameLift Servers.

Mit dieser neuen Funktion müssen Spielsitzungen behoben werden, die zwar aktiv sind, sich aber in einem schlechten Zustand befinden, wodurch Computerressourcen daran gehindert werden, neue Spielsitzungen zu hosten. Bisher mussten Kunden per Fernzugriff auf den Computer zugreifen, um eine Spielsitzung manuell zu beenden.

Sie haben zwei Kündigungsmethoden zur Auswahl. Bei der ersten Methode wird versucht, eine Spielsitzung mithilfe der benutzerdefinierten Shutdown-Sequenz, die Aktionen zur Benachrichtigung der Spieler und zur Auflösung von Spieldaten beinhalten kann, ordnungsgemäß zu beenden. Bei der zweiten Methode wird der Serverprozess gestoppt, wodurch die Spielsitzung sofort beendet wird. Diese zweite Methode stellt sicher, dass die Spielsitzung auch dann endet, wenn der Serverprozess nicht reagiert.

Weitere Informationen:

- [Eine Spielsitzung mithilfe der Amazon GameLift Servers Konsole beenden](#), Amazon GameLift ServersEntwicklerhandbuch
- [TerminateGameSession](#), Amazon GameLift ServersAPI-Referenz

19. Dezember 2024: Amazon GameLift Servers veröffentlicht Game-Engine-Plugin-Unterstützung für Managed Containers

Aktualisierte Plugin-Versionen:

Amazon GameLift ServersPlugin für Unreal Engine, Version 2.0.0

- Es wurde aktualisiert, um das C++-Server-SDK 5.2.0 mit Unterstützung für verwaltete Container zu unterstützen.
- Unterstützung für Unreal Engine 5.4 und 5.5 hinzugefügt.

Amazon GameLift ServersPlugin für Unity, Version 3.0.0

- Es wurde aktualisiert, um das C++-Server-SDK 5.2.0 mit Unterstützung für verwaltete Container zu unterstützen.
- Support für Unity 2021.3 LTS und 2022.3 LTS für Windows und Mac OS.

Das Amazon GameLift Servers Plugin für die Game-Engines Unreal und Unity bietet Tools und Workflows, die deine Schritte zur Inbetriebnahme eines Spiels vereinfachen. Amazon GameLift Servers Amazon GameLift Servers ist ein vollständig verwalteter Cloud-Hosting-Dienst, mit dem Spieleentwickler dedizierte Spieleserver für sitzungsbasierte Multiplayer-Spiele verwalten und skalieren können.

Die neuesten Plugin-Versionen bieten die folgenden Verbesserungen:

- Geführter Workflow für das Hosting mit Managed Containern. Dieser Workflow führt Sie durch die Schritte zur Einrichtung eines Container-Images mit Ihrer Gameserver-Software und zur Bereitstellung einer cloubasierten Hosting-Lösung für Ihren Spielservers. Der Workflow bietet zwei verschiedene Einsatzszenarien: eine einfache Bereitstellung und eine vollständigere Bereitstellung mit einer Warteschlange für die Platzierung von Spielsitzungen und einem FlexMatch Matchmaker. In jedem Szenario werden Amazon GameLift Servers Containerflotten und unterstützende AWS Ressourcen generiert.
- Verbesserter Prozess für die Einrichtung von AWS Benutzerprofilen und die Verwaltung der AWS Zugangsdaten für die Verwendung von Plug-ins. Sie können mehrere Profile verwalten, um mit unterschiedlichen AWS Konten, Kontonutzern und Regionen zu arbeiten.
- Zusätzliche Funktionen zur Aktualisierung vorhandener Containerflotten. Du kannst neue Container-Images bereitstellen (z. B. für Updates der Spielserverversionen) und die Flottenkonfigurationseinstellungen ändern, ohne von vorne beginnen zu müssen.
- Verbesserte Workflows für das Hosting mit Amazon GameLift Servers Anywhere-Flotten und Managed EC2. Zu den Verbesserungen, die auf Kundenfeedback basieren, gehören eine bessere Anleitung mit Tipps und Links zu hilfreichen Ressourcen.

In den Bereitstellungsszenarien für Managed Containers und Managed EC2-Lösungen werden AWS CloudFormation Vorlagen verwendet, um die AWS Ressourcen für jedes Szenario zu erstellen und bereitzustellen. Diese Vorlagen sind im Amazon GameLift Servers Plugin-Download enthalten und können bearbeitet werden. Du kannst sie unverändert verwenden oder sie für dein Spiel modifizieren.

Weitere Informationen:

- [Plugin für Unreal: Setze dein Spiel auf einer verwalteten Containerflotte ein](#), Amazon GameLift ServersEntwicklerhandbuch
- [Plugin für Unity: Stellen Sie Ihr Spiel in einer verwalteten Containerflotte bereit](#), Amazon GameLift ServersEntwicklerhandbuch
- [Laden Sie das Plugin für Unreal von herunter GitHub](#)
- [Laden Sie das Plugin für Unity von herunter GitHub](#)
- [Spiele-Hosting mit Amazon GameLift Servers](#)
- [Amazon GameLift Servers-Forum](#)

12. November 2024: Amazon GameLift Servers Startet Multiplayer-Game-Hosting mit Managed Containers

Aktualisierte SDK-Versionen:

- AWS SDK 1.11.445
- Server-SDK, Version 5.2.0 (alle Sprachen)

Amazon GameLift Servers veröffentlicht zur allgemeinen Verfügbarkeit eine neue Hosting-Lösung für containerisierte Spielserver-Workloads. Mit dieser Version können Spieleentwickler nun die Vorteile der Containerisierung nutzen, darunter konsistente, sichere Umgebungen, einen vereinfachten Bereitstellungsprozess und eine optimierte Ressourcennutzung.

Verwaltete Containerflotten verwenden Amazon EC2 EC2-Instances, die von in Ihrem Namen und Amazon GameLift Servers auf der Grundlage Ihrer Konfigurationen verwaltet werden. Sie erstellen eine benutzerdefinierte Container-Architektur für Ihr Spiel und stellen Container-Images bereit, indem Sie sie in einem Amazon Elastic Container Registry (Amazon ECR) -Repository speichern. Containerflotten sind nur für Linux-basierte Spielserver verfügbar. Spielserver müssen in das Server-SDK 5.2.0 oder höher integriert sein.

Mit verwalteten Containerflotten erhalten Sie dieselben Vorteile wie mit verwalteten EC2-Flotten. Dazu gehören Unterstützung für On-Demand-Instance-Typen und Spot-Instance-Typen, intelligente Kapazitätsskalierung, Platzierung von Spielsitzungen mit Warteschlangen und Matchmaking. Sie erhalten außerdem dieselben Kennzahlen wie bei anderen Flottenarten sowie einige neue Kennzahlen für Container. Zu den weiteren Funktionen für Containerflotten gehören:

- Ausrichtung auf serverlose Lösungen für containerisierte Workloads Führen Sie einen Spielserver-Prozess pro Container aus und packen Sie viele Container auf jede Flotteninstanz, um die Ressourcen optimal zu nutzen. Wenn du lieber Container mit mehreren Gameserver-Prozessen haben möchtest, kannst du den Amazon GameLift Servers Agenten für die automatisierte Hostverwaltung verwenden.
- Optimierte Flottenerstellung. Containerflotten sind so konzipiert, dass sie minimale Einrichtungseinstellungen mit sinnvollen suggested/default Werten erfordern. Sie können schnell eine funktionierende Flotte bereitstellen und dann individuelle Einstellungen nach Bedarf anpassen.

- Versionierungstools für die Container-Architektur. Sie können jetzt eine Container-Gruppendefinition aktualisieren (was einer Container-"Aufgabe" ähnelt), mehrere Versionen verwalten und angeben, welche Version für eine Flotte bereitgestellt werden soll.
- Tools zur Flottenaktualisierung. Mit Containerflotten musst du keine neue Flotte mehr erstellen, wenn du ein Versionsupdate für den Spielservers veröffentlichen möchtest. Stattdessen kannst du jetzt dein Container-Image aktualisieren und die Updates für bestehende Flotten bereitstellen.

Sie können Amazon GameLift Servers Containerflotten überall dort aufbauen, AWS-Region wo Flotten mit mehreren Standorten Amazon GameLift Servers unterstützt werden, und Sie können Container-Flotteninstanzen an jedem unterstützten Remote-Standort bereitstellen. Weitere Details finden Sie unter [Amazon GameLift Servers Servicestandorte](#). Managed Containers ist derzeit in AWS China Regionen nicht verfügbar.

Weitere Informationen:

- Blogbeitrag: [Nutzen Sie vollständig verwaltete Container, um Multiplayer-Spiele auf globaler Ebene zu hosten Amazon GameLift Servers](#)
- [Verwaltete Container](#) Überblick, Amazon GameLift Servers Entwicklerhandbuch
- [So funktionieren Container in Amazon GameLift Servers](#), Amazon GameLift Servers Entwicklerhandbuch
- [Entwicklungs-Roadmap für Hosting mit Amazon GameLift Servers verwalteten Containern](#), Amazon GameLift Servers Entwicklerhandbuch
- [CreateContainerFleet](#), Amazon GameLift Servers API-Referenz

19. September 2024: Amazon GameLift Servers veröffentlicht ein Update für das C++-Server-SDK und die Plugins für Unreal Engine

Aktualisierte SDK-Versionen:

C++ Server-SDK, Version 5.1.3

- Neue Protokollierungsfunktionen. Sie können jetzt auf SDK-Anforderungsprotokolle zugreifen.
- Die Zuverlässigkeit der SDK-Nachrichtenübertragung wurde verbessert. Das SDK verwendet jetzt robustere Wiederverbindungsmechanismen für die Wiederherstellung bei Netzwerkunterbrechungen oder zufälligen Nachrichtenverlusten.

Aktualisierte Plugin-Versionen:

Amazon GameLift ServersPlugin für Unreal Engine, Version 1.1.2

- Es wurde aktualisiert, um die neueste Version des C++-Server-SDK 5.1.3 zu unterstützen.
- Wenn Sie im Amazon GameLift Servers Plugin für Unreal Engine nach einer ausführbaren Server-Build-Datei für eine Flotte suchen, haben Sie jetzt die Möglichkeit, alle Dateien zu durchsuchen.

C++-Server-SDK-Plugin für Unreal, Version 5.1.2

- Es wurde aktualisiert, um die neueste Version des C++-Server-SDK 5.1.3 zu unterstützen.

Weitere Informationen:

- [Integration von Spielen mit dem Amazon GameLift Servers Plugin für Unreal Engine, Entwicklerhandbuch Amazon GameLift Servers](#)
- [Amazon GameLift ServersPlugin- und SDK-Downloads](#)

5. September 2024: Amazon GameLift Servers verbessert die Beobachtbarkeit des Flottenerstellungsprozesses

Auf der Grundlage von Kundenfeedback haben wir den Amazon GameLift Servers Arbeitsablauf für die Erstellung einer verwalteten EC2-Flotte und deren Vorbereitung für die Ausrichtung von Spielsitzungen geklärt. Zu den Verbesserungen gehören:

- Wir haben genauere und genauere Beschreibungen der einzelnen Phasen des Flottenerstellungsprozesses bereitgestellt. Diese verbesserte Transparenz macht es einfacher, Probleme schneller zu lokalisieren und zu lösen.
- In den Phasen Aufbau und Aktivierung wurden Aufgaben zur Instanzbereitstellung (Aufbau) besser von Aufgaben zum Starten von Spielserverprozessen und zum Herstellen einer Verbindung mit dem Amazon GameLift Servers Dienst (Aktivierung) getrennt. Diese Änderung macht es einfacher, die wahrscheinliche Ursache von Problemen zu erkennen. Darüber hinaus können Sie jetzt per Fernzugriff eine Verbindung zu Flotten herstellen, wenn diese sich in der Aktivierungsphase befinden.
- Zwei neue Ereignisse zur Flottenerstellung informieren über den Erfolg oder Misserfolg von Installationsskripten für Spieleserver. Wenn dein Gameserver-Build ein Installationsskript enthält,

Amazon GameLift Servers versucht es auszuführen und löst eines der folgenden neuen Ereignisse aus:

- FLEET_CREATION_COMPLETED_INSTALLER
- FLEET_CREATION_FAILED_INSTALLER

Weitere Informationen:

- [Debuggen von Amazon GameLift Servers-Flottenproblemen](#), Amazon GameLift ServersEntwicklerhandbuch
- [Datentyp des Ereignisses](#), Amazon GameLift ServersAPI-Referenz

15. August 2024: Amazon GameLift Servers verbessert das Konsolenerlebnis

Basierend auf Kundenfeedback haben wir die folgenden Aktualisierungen am [Amazon GameLift ServersKonsolenerlebnis](#) vorgenommen:

- Ihre Anzeigeeinstellungen für Seiten werden jetzt automatisch für Ihren AWS Kontonutzer gespeichert und angewendet, wenn Sie zur Seite zurückkehren. Mit den Anzeigeeinstellungen können Sie auswählen, welche Informationen in einer Tabelle angezeigt werden sollen, z. B. auf der Seite mit der Liste der Flotten. Passen Sie Ihre Anzeigeeinstellungen an, indem Sie das



Symbol in der oberen rechten Ecke einer Tabelle verwenden.

- Der Workflow Create Fleet für verwaltete EC2-Flotten wurde optimiert, um die Auswahl von Flottenstandorten und Instance-Typen zu kombinieren. Wir haben es Ihnen leichter gemacht, den richtigen Instance-Typ für Ihre Flotte zu finden, auch wenn Sie Ihre Standortauswahl ändern.

Weitere Informationen:

- [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#), Amazon GameLift ServersEntwicklerhandbuch

25. Juli 2024: Amazon GameLift Servers fügt Unterstützung für die AWS Nigeria Local Zone hinzu

Mit Amazon GameLift Servers verwaltetem Hosting können Sie jetzt Spieleserverressourcen in Nigeria und Westafrika bereitstellen und die Reichweite Ihrer Spiele auf Spieler in ganz Afrika ausdehnen. Verwende AWS Local Zones, um Spieleserver geografisch näher an deinen Spielern zu platzieren, um die Latenz zu reduzieren und das Gameplay deutlich zu verbessern.

Um sofort mit der Ausrichtung von Spielsitzungen in Nigeria zu beginnen, fügen Sie die neue Nigeria Local Zone als abgelegenen Standort zu einer neuen oder bestehenden Flotte mit mehreren Standorten hinzu. Falls euer Spiel dies nutzt Amazon GameLift ServersFlexMatch, aktualisiert eure Flotten in eurer Spielerzuweisungswarteschlange, sodass sie auch die neue lokale Zone enthalten. Bei Flotten mit mehreren Standorten kannst du die Hosting-Kapazitäten an jedem Standort direkt verwalten.

Die übergeordnete Region AWS-Region für die lokale Zone Lagos, Nigeria, ist die Region Afrika (Kapstadt) (af-south-1), die Amazon GameLift Servers auch als abgelegener Standort unterstützt wird. Der Name der lokalen Zone in Nigeria lautet af-south-1-los-1.

Weitere Informationen:

- [Amazon GameLift ServersServicestandorte](#), Amazon GameLift ServersEntwicklerhandbuch
- [Flottenstandorte aktualisieren](#), Amazon GameLift ServersEntwicklerhandbuch

02. Juli 2024: Amazon GameLift Servers veröffentlicht ein neues Konsolentool zum Anzeigen von Spiellersitzungsdaten

Die Amazon GameLift Servers Konsole bietet jetzt ein Tool zur Suche nach Spiellersitzungen, mit dem Sie Sitzungsinformationen nach Spielsitzungs-ID, Spiellersitzungs-ID oder Spieler-ID abrufen können. Spiele, die FlexMatch Spiellersuche verwenden, generieren automatisch Spiellersitzungen für jeden passenden Spieler. Für alle anderen Spiele sind Spiellersitzungen eine optionale Funktion.

Sie finden das Tool zur Suche nach Spiellersitzungen in der Hauptnavigation der Amazon GameLift Servers Konsole. Sieh dir einzelne Spiellersitzungen an oder vergleiche die Daten mehrerer Spiellersitzungen. Sie können die Sitzungsdaten von Spielern auch öffnen, wenn Sie sich die Detailseite einer Spielsitzung ansehen.

Weitere Informationen:

- [Spiel- und Spielersitzungen in der Amazon GameLift Servers Konsole](#), Amazon GameLift ServersEntwicklerhandbuch

24. April 2024: Amazon GameLift Servers veröffentlicht eine Vorschau auf Containerflotten

Amazon GameLift Servers bietet jetzt eine Vorschau auf Containerflotten, die Ihnen verbesserte Portabilität, Skalierbarkeit, Fehlertoleranz und Agilität bieten.

In Containerflotten hosten Amazon EC2 EC2-Instances einen oder mehrere Ihrer Container. Zu diesen Containern gehören Ihr Spieleserver sowie alles, was er benötigt, einschließlich Abhängigkeiten und Konfigurationen. Zu den Abhängigkeiten gehören beispielsweise SDKs Softwarepakete. Nachdem Sie Ihren Container in Ihr privates Amazon Elastic Container Registry hochgeladen haben, Amazon GameLift Servers füllt er Ihre Flotte mit dem Container.

Um in einer Containerflotte zu funktionieren, muss Ihr Spieleserver unter Linux laufen und in das Server SDK 5.x integriert sein. In einer Containerflotte haben Sie eine fein abgestimmte Kontrolle über die Hosting-Ressourcen, sodass Sie den Verbrauch von Ressourcen wie CPU-Einheiten und Arbeitsspeicher optimieren können. Sie können auch mehrere Spieleserver in einem Container hosten, um den Ressourcenverbrauch zu reduzieren.

In einer Containerflotte erhalten Sie viele der gleichen Vorteile wie andere Flotten, wie z. B. On-Demand-Instance-Typen, Skalierung (automatisch und manuell), Warteschlangen und Spielersuche. Sie erhalten außerdem dieselben Kennzahlen wie bei anderen Flottenarten, zusätzlich zu einigen neuen Kennzahlen für Container. Mit Containerflotten erreichen Sie Spieler an folgenden Standorten und Regionen auf globaler Ebene:

- ap-northeast-1
- ap-northeast-2
- ap-southeast-2
- eu-central-1
- eu-west-1
- us-east-1
- us-west-2

Um noch mehr Regionen und lokale Zonen zu erreichen, erstellen Sie Containerflotten mit mehreren Standorten.

Weitere Informationen:

- [Hosting-Verwaltung mit Amazon GameLift Servers Containern, Entwicklerhandbuch Amazon GameLift Servers](#)
- [CreateContainerGroupDefinition](#), Amazon GameLift ServersAPI-Referenz

13. Februar 2024: Amazon GameLift Servers führt Verbesserungen am Amazon GameLift Servers Plugin für Unreal Engine ein und vereinfacht die Installation SDKs

Aktualisierte SDK-Versionen:

- Go Server SDK, Version 5.1.0
- C# Server-SDK, Version 5.1.2
- C++-Server-SDK, Version 5.1.2

Wir haben die folgenden Verbesserungen vorgenommen:

- Die Zuverlässigkeit des SDK wurde verbessert, indem eine automatische Wiederverbindung im Falle einer Netzwerkunterbrechung hinzugefügt wurde.
- [Go] Sie können jetzt `InitSDK()` mit oder ohne Serverparameter aufrufen. Spieleserver, die auf Amazon GameLift Servers verwalteten EC2-Flotten laufen, lesen die Serverparameter direkt aus Umgebungsvariablen. Spieleserver auf Amazon GameLift Servers Anywhere-Flotten müssen `InitSDK()` mit Serverparametern aufgerufen werden.

Aktualisierte Plugin-Versionen:

- Amazon GameLift ServersPlugin für Unreal Engine, Version 1.1.0
- Amazon GameLift ServersPlugin für Unity, Version 2.1.0
- C++-Server-SDK-Plugin für Unreal, Version 5.1.1
- C# Server-SDK-Plugin für Unity, Version 5.1.2

Wir haben die folgenden Verbesserungen vorgenommen:

- [Amazon GameLift ServersPlugin für Unreal Engine] Die Installationsanweisungen wurden aktualisiert und die Paketierung vereinfacht. Dieses Plugin enthält jetzt die neueste Version des C++ Server SDK für Unreal.
- Die Plugins wurden aktualisiert, sodass sie die neueste Version des Server-SDK für Amazon GameLift Servers unterstützen.

Weitere Informationen:

- [Integration von Spielen mit dem Amazon GameLift Servers Plugin für Unreal Engine](#), Amazon GameLift ServersEntwicklerhandbuch
- [Amazon GameLift ServersPlugin- und SDK-Downloads](#)

14. Dezember 2023: Amazon GameLift Servers Fügt die Möglichkeit hinzu, die Spieleigenschaften aktiver Spielsitzungen zu aktualisieren

Du konntest bereits beim Erstellen von Spielsitzungen Spieleigenschaften festlegen und Spielsitzungen nach bestimmten Eigenschaften durchsuchen. Jetzt kannst du diese Eigenschaften auch in einer aktiven Spielsitzung hinzufügen und aktualisieren.

Zum Beispiel stimmen deine Spieler auf einer Karte ab, auf der sie spielen möchten. Ihr Spielclient ruft `UpdateGameSession` auf, um einen `GameProperty` Wert zu ändern `{"Key": "map", "Value": "jungle"}`. Ihr Spiel implementiert dann die neue Map für die Spieler in der Spielsitzung.

Mithilfe des `SearchGameSessions` Vorgangs können Spieladministratoren auch nützliche Daten aus den Spieleigenschaften abrufen. Administratoren können beispielsweise Spielsitzungen auflisten, die den Status Wert `ACTIVE` und diese Spieleigenschaft haben: `{"Key": "map", "Value": "desert"}`.

Weitere Informationen:

- [the section called “Integrieren Sie die Funktionalität des Spielclients”](#), Amazon GameLift ServersEntwicklerhandbuch
- [GameProperty](#), Amazon GameLift ServersAPI-Referenz
- [UpdateGameSession](#), Amazon GameLift ServersAPI-Referenz
- [SearchGameSessions](#), Amazon GameLift ServersAPI-Referenz

21. November 2023: Amazon GameLift Servers startet Unterstützung für Infrastructure-as-Code-Tools wie Terraform und Pulumi, unterstützt von AWS -Cloud-Control- API

Sie können jetzt Ihren gesamten Amazon GameLift Servers Ressourcenstapel mithilfe von Infrastructure as Code (IaC) -Tools verwalten. Zu diesen Tools gehören AWS CloudFormation auch Tools von Drittanbietern wie Terraform und Pulumi. Mit dieser zusätzlichen Unterstützung können Sie sich jetzt auf die Entwicklung Ihres Spiels konzentrieren und DevOps Strategien nutzen, um sich um das Ressourcenmanagement, CI/CD und die Bereitstellung für Ihre Kunden zu kümmern.

Sie können jetzt auch alle Amazon GameLift Servers Ressourcentypen mithilfe der AWS Cloud Control API bereitstellen und konfigurieren. Sie können weiterhin mit Ressourcen arbeiten, indem Sie die Amazon GameLift Servers APIs oder die AWS CloudFormation Vorlagen für verwenden Amazon GameLift Servers.

Einzelheiten zu den über IaC verfügbaren Amazon GameLift Servers Ressourcen finden Sie in der [Amazon GameLift Servers Ressourcentyp-Referenz](#) zur Amazon GameLift Servers Ressourcentyp-Referenz.

Darüber hinaus können Sie Ihre Flotten jetzt mithilfe von AWS CloudFormation Vorlagen oder der AWS Cloud Control-API mithilfe der neuen [Fleet-Eigenschaft](#) automatisch skalieren: `ScalingPolicies`

Die Cloud Control API bietet Entwicklern einen Standardsatz APIs zum Erstellen, Lesen, Aktualisieren, Löschen und Auflisten von Ressourcen (CRUDL) für Hunderte von AWS Diensten und mehrere Tools von Drittanbietern wie Terraform und Pulumi.

Weitere Informationen:

- [AWS CloudFormation](#)
- [AWS Cloud Control API](#)
- [AWS CC Terraform-Anbieter](#)
- [Pulumi](#)

16. November 2023: Amazon GameLift Servers aktualisiert das eigenständige Plugin für Unity

Aktualisierte SDK-Versionen: Amazon GameLift Servers Plugin für Unity, Version 2.0.0

Das Amazon GameLift Servers Plugin für Unity bietet Tools und Workflows, mit denen Sie die Schritte zur Einrichtung Ihres Unity-Spiels für Cloud-Hosting optimieren können. Amazon GameLift Servers Amazon GameLift Servers ist ein vollständig verwalteter Dienst, mit dem Spieleentwickler spezielle Spielservers für sitzungsbasierte Multiplayer-Spiele verwalten und skalieren können.

Mit dieser Version wurde das Plugin für Unity aktualisiert und nutzt nun die neuesten Amazon GameLift Servers Funktionen, darunter Server-SDK Version 5.x und Unterstützung für lokale Tests mit Anywhere. Amazon GameLift Servers Das Plugin ist mit den Unity-Versionen Unity 2021.3 LTS und 2022.3 LTS kompatibel.

Zu den wichtigsten Funktionen des Plugins gehören:

- Geführte UI-Workflows im Unity-Editor für die folgenden Szenarien:
 - Testen Sie Ihre Spieleintegration, Amazon GameLift Servers indem Sie Ihre lokale Workstation als Host verwenden. Dieser Workflow hilft dir dabei, eine Amazon GameLift Servers Anywhere-Flotte für deinen lokalen Computer einzurichten, Instanzen deines Spielservers und Clients zu starten, eine Spielsitzung Amazon GameLift Servers anzufordern und dem Spiel beizutreten.
 - Stellen Sie eine Cloud-Hosting-Lösung für Ihren integrierten Spielservers mit Amazon GameLift Servers verwaltetem EC2 und unterstützenden AWS Ressourcen bereit. Dieser Workflow hilft Ihnen bei der Konfiguration Ihres Spiels für Cloud-Hosting und bietet drei Bereitstellungsszenarien:
 - Setze den Spielservers auf einer einzigen Flotte ein.
 - Setze den Spielservers für eine Reihe kostengünstiger Spot-Flotten in mehreren AWS Regionen ein.
 - Stellen Sie den Spielservers mit einem FlexMatch Matchmaker bereit.
- Möglichkeit, Benutzerprofile einzurichten, die auf einen AWS Kontobenutzer verweisen, und eine AWS Standardregion festzulegen. Sie können mehrere Profile verwalten, um in verschiedenen AWS Konten, Kontonutzern und Regionen zu arbeiten.
- Besondere Annehmlichkeiten, die dazu beitragen, die Amazon GameLift Servers Integrations- und Bereitstellungsprozesse zu rationalisieren, darunter:
 - Jede Hosting-Lösung umfasst unterstützende AWS Ressourcen, darunter einen Amazon Cognito Cognito-Benutzerpool, der eine einzigartige Spieler IDs - und Spielvalidierung ermöglicht. Die Lösungen umfassen auch einen Amazon S3 S3-Bucket für Speicher, Amazon SNS SNS-Ereignisbenachrichtigungen, AWS Lambda Funktionen und andere Ressourcen.
 - Für den Anywhere-Workflow automatisiert das Plugin die erforderlichen Serverparametereinstellungen.

- Für den Amazon EC2 EC2-Workflow bietet jede Bereitstellungslösung einen integrierten Client-Backend-Service, der Lambda-Funktionen verwendet. Der Backend-Service befindet sich zwischen dem Spielclient und dem Amazon GameLift Servers Service und verwaltet alle direkten Aufrufe des Dienstes. Amazon GameLift Servers
- Inhalt für Integrationstests, einschließlich Ressourcen und Code für ein einfaches Beispiel-Multiplayer-Spiel zur Veranschaulichung der Integration von Spieleserver und Spielclient.
- Plugin-Dokumentation mit ausführlichen Integrationsanleitungen und Beispielcode.

In allen Bereitstellungsszenarien, auch für Anywhere- und Amazon EC2 EC2-Flotten, werden AWS CloudFormation Vorlagen verwendet, um die AWS Ressourcen für die Lösung Ihres Spiels zu beschreiben und bereitzustellen. Diese Vorlagen sind im Amazon GameLift Servers Plugin-Download enthalten. Du kannst sie so verwenden, wie sie sind, oder sie für dein Spiel anpassen.

Weitere Informationen:

- [Amazon GameLift ServersPlugin für Unity \(Server-SDK 5.x\)](#), Amazon GameLift ServersEntwicklerhandbuch
- [Laden Sie das Plugin herunter von GitHub](#)
- [Über das Amazon GameLift Servers Hosting](#)
- [Amazon GameLift Servers-Forum](#)

2. November 2023: Amazon GameLift Servers Fügt Unterstützung für gemeinsame Anmeldeinformationen hinzu

Aktualisierte SDK-Versionen: AWS SDK 1.11.193

Die neue Funktion für Amazon GameLift Servers gemeinsame Anmeldeinformationen ermöglicht es Anwendungen, die auf verwalteten EC2-Flotten bereitgestellt werden, mit anderen Ressourcen zu interagieren. AWS Dieses Update betrifft Anwendungen, die Sie zusammen mit Gameserver-Binärdateien bündeln und bereitstellen, die in das Server-SDK Version 5.x oder höher integriert sind. (Programmdateien von Spieleservern können bereits mithilfe der Server-SDK 5.x-Aktion Anmeldeinformationen anfordern.) `GetFleetRoleCredentials()`

Wenn Sie beispielsweise Ihren Gameserver-Build mit einem CloudWatch Amazon-Agenten bereitstellen möchten, um EC2-Instance-Metriken und andere Daten zu sammeln, benötigt der Agent die Erlaubnis, mit Ihren CloudWatch Ressourcen zu interagieren. Dazu müssen Sie zunächst

eine AWS Identity and Access Management IAM-Rolle mit Berechtigungen zur Nutzung der CloudWatch Ressourcen einrichten und dann eine Flotte konfigurieren, für die die IAM-Rolle und die gemeinsamen Anmeldeinformationen aktiviert sind. Wenn Sie Amazon GameLift Servers Ihren Spielserver-Build auf jeder EC2-Instance bereitstellen, generiert er eine Datei mit gemeinsamen Anmeldeinformationen und speichert sie auf der Instance. Alle Anwendungen auf der Instance können die gemeinsamen Anmeldeinformationen verwenden. Amazon GameLift Servers aktualisiert die temporären Anmeldeinformationen während der gesamten Lebensdauer der Instanz automatisch.

Sie können gemeinsame Anmeldeinformationen aktivieren, wenn Sie eine verwaltete EC2-Flotte mit den folgenden Methoden erstellen:

- Im Workflow zur Erstellung von Amazon GameLift Servers Konsolenflotten.
- Beim Aufrufen der Service-API-Operation `CreateFleet` mit dem neuen Parameter `InstanceRoleCredentialsProvider`.
- Beim Aufrufen der AWS CLI-Operation `aws gamelift create-fleet` mit dem Parameter `instance-role-credentials-provider`.

Weitere Informationen:

- [Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten](#), Amazon GameLift Servers Entwicklerhandbuch
- [CreateFleet, InstanceRoleCredentialsProvider](#), Amazon GameLift Servers API-Referenz
- [Richten Sie eine IAM-Servicerolle](#) ein, Amazon GameLift Servers Entwicklerhandbuch

28. September 2023: Amazon GameLift Servers veröffentlicht ein neues eigenständiges Plugin für Unreal Engine Engine

Aktualisierte SDK-Versionen: Amazon GameLift Servers Plugin für Unreal Engine Version 1.0.0

Das Amazon GameLift Servers Plugin für Unreal Engine bietet Tools und Workflows, die deine Schritte zur Inbetriebnahme eines Spiels für das Cloud-Hosting vereinfachen. Amazon GameLift Servers ist ein vollständig verwalteter Dienst, mit dem Spieleentwickler dedizierte Spielserver für sitzungsbasierte Multiplayer-Spiele verwalten und skalieren können. Das Plugin unterstützt die UE-Versionen 5.0, 5.1 und 5.2. Zu den wichtigsten Features gehören:

- Geführte Benutzeroberflächenworkflows im Unreal Editor führen Sie durch die folgenden Pfade:

- Testen Sie Ihre Spielintegration, Amazon GameLift Servers indem Sie Ihre lokale Workstation als Host verwenden. Dieser Workflow hilft Ihnen dabei, eine Amazon GameLift Servers Anywhere-Flotte für Ihren lokalen Computer einzurichten, Instanzen Ihres Spieleservers und Clients zu starten, eine Spielsitzung Amazon GameLift Servers anzufordern und Verbindungsinformationen für die neue Spielsitzung abzurufen.
- Stellen Sie eine Amazon EC2 EC2-Cloud-Hosting-Lösung für Ihren integrierten Spieleserver bereit. Dieser Workflow hilft Ihnen bei der Konfiguration Ihres Spiels für Cloud-Hosting und bietet drei verschiedene Einsatzszenarien: Bereitstellung auf einer einzelnen Flotte, Bereitstellung auf einer Reihe von Spot-Flotten in mehreren Regionen oder Bereitstellung auf einer Gruppe von Flotten mit einem Matchmaker. FlexMatch Die Lösung für jedes Einsatzszenario umfasst Amazon GameLift Servers Ressourcen und unterstützende Ressourcen. AWS
- Möglichkeit, Benutzerprofile einzurichten, die mit einem AWS Kontobutzer verknüpft sind, und eine AWS Standardregion zu definieren. Sie können mehrere Profile verwalten, um in verschiedenen AWS Konten, Kontonutzern und Regionen zu arbeiten.
- Besondere Annehmlichkeiten, die dazu beitragen, die Amazon GameLift Servers Integrations- und Bereitstellungsprozesse zu rationalisieren, darunter:
 - Jede Hosting-Lösung umfasst unterstützende AWS Ressourcen, darunter einen grundlegenden Amazon Cognito Cognito-Benutzerpool, der einen einzigartigen Player bietet IDs, einen Amazon S3-Bucket für Speicher, Amazon SNS-Ereignisbenachrichtigungen und AWS Lambda Funktionen.
 - Für den Anywhere-Workflow automatisiert das Plugin die erforderlichen Serverparametereinstellungen mithilfe von Befehlszeilenargumenten.
 - Für den Amazon EC2 EC2-Workflow bietet jede Bereitstellungslösung einen integrierten Client-Backend-Service, der Lambda-Funktionen verwendet. Der Backend-Service empfängt Anfragen von Spieleclients und leitet sie an den Service weiter. Amazon GameLift Servers
- Inhalte für Integrationstests, darunter eine Starter-Spielkarte und zwei Testkarten mit grundlegenden Entwürfen und Benutzeroberflächenelementen.
- Plugin-Dokumentation mit ausführlichen Integrationsanleitungen und Beispielcode.

In allen Bereitstellungsszenarien, auch für Anywhere- und Amazon EC2 EC2-Flotten, werden AWS CloudFormation Vorlagen zur Beschreibung der Lösungen verwendet. Das Plugin verwendet diese Vorlagen bei der Bereitstellung von Amazon GameLift Servers Ressourcen für Ihr Spiel. Diese Vorlagen sind im Amazon GameLift Servers Plugin-Download enthalten und können bearbeitet werden. Du kannst sie unverändert verwenden oder sie für dein Spiel modifizieren.

Weitere Informationen:

- [Amazon GameLift ServersPlugin für Unreal Engine](#), Amazon GameLift ServersEntwicklerhandbuch
- [Laden Sie das Plugin herunter von GitHub](#)
- [Über das Amazon GameLift Servers Hosting](#)
- [Amazon GameLift Servers-Forum](#)

17. August 2023: Amazon GameLift Servers bietet Gameserver-Hosting mit AWS Graviton-Prozessoren

Aktualisierte SDK-Versionen: AWS SDK 1.11.144

Mit können Amazon GameLift Servers Sie Ihre Spiele jetzt mithilfe von EC2-Instances mit AWS Graviton-Prozessoren in der Cloud hosten. Graviton-Instances wurden AWS mit ARM64-basierten Prozessoren entwickelt und bieten das beste Preis-Leistungs-Verhältnis für Cloud-Workloads mit EC2, mit einer Verbesserung von bis zu 40% gegenüber vergleichbaren x86-basierten Instances. Die neuesten Graviton3-Prozessoren bieten eine um bis zu 25% bessere Rechenleistung als frühere Versionen.

Mit Amazon GameLift Servers können Sie jetzt aus diesen neuen Instanzen der AWS Graviton-Familie wählen:

- Graviton2-basierte Instanzen: c6g, c6gn, r6g, m6g, g5g
- Graviton3-basierte Instanzen: c7g, r7g, m7g

Weitere Informationen:

- [AWS Graviton-Prozessor: Erfahren Sie mehr über die Vorteile](#) und praktischen Einsatzmöglichkeiten von Graviton-basierten EC2-Instances.
- [Erste Schritte mit Graviton: Verschaffen](#) Sie sich einen Überblick über die Graviton-basierten Instances und erfahren Sie, wie Anwendungen auf ihnen je nach Betriebssystem, Sprache und Laufzeit ausgeführt werden.

Note

Graviton Arm-Instances erfordern einen Server-Build für ein Linux-AMI. Server-SDK 5.1.1 oder neuer ist für C++ und C# erforderlich. Server-SDK 5.0 oder neuer ist für Go erforderlich.

Diese Instances bieten keine out-of-the-box Unterstützung für die Mono-Installation auf Amazon Linux 2023 (AL2023) oder Amazon Linux 2 (AL2).

27. Juli 2023: Amazon GameLift Servers veröffentlicht das Server-SDK 5.1.0 mit zusätzlicher Unterstützung für die Unity-Entwicklung

Aktualisierte SDK-Versionen: Server-SDK SDK for C++, C#/Unity, Unreal 5.1.0

Die neueste Version des Amazon GameLift Servers Server-SDK enthält Updates für C++, C# und das Unreal-Plugin sowie ein neues Plugin zur Verwendung mit der Unity-Game-Engine. Spieleentwickler integrieren das Amazon GameLift Servers Server-SDK in Spieleserver, auf denen sie es bereitstellen. Amazon GameLift Servers

Die neueste Server-SDK-Version enthält die folgenden Updates, die eine Reihe von Kundenanfragen beinhalten:

- Sprachspezifische SDK-Pakete herunterladen — Die aktualisierte [Amazon GameLift Servers Download-Website](#) enthält SDK-Pakete für jede Sprache. Sie können aktuelle oder frühere Versionen herunterladen.
- Neues C#-Server-SDK-Plugin für Unity — Das neue Server-SDK-Paket für Unity enthält fertige C#-Bibliotheken, die Sie mit dem Paketmanager im Unity Editor installieren können (siehe den neuen [Unity-Integrationsleitfaden](#)). Diese Bibliotheken enthalten die erforderlichen Abhängigkeiten durch. UnityNuGet Sie können dieses Plugin mit Unity 2020.3 LTS, 2021.3 LTS und 2022.3 LTS für Windows und Mac OS verwenden. Es unterstützt die Profile .NET Framework und .NET Standard von Unity mit .NET Standard 2.1 und .NET 4.x.
- Konsolidierte .NET-Lösung für C# — Das Server-SDK für C# unterstützt jetzt .NET Framework 4.6.2 (aktualisiert von 4.6.1) und .NET 6.0 in einer einzigen Lösung. .NET Standard 2.1 ist mit den von Unity erstellten Bibliotheken verfügbar.
- Server-SDK 5.1.0 aktualisiert
 - [C++, C#, Unreal] Sie können jetzt `InitSDK()` mit oder ohne Serverparameter aufrufen. Spieleserver, die auf Amazon GameLift Servers verwalteten EC2-Flotten laufen, lesen die Serverparameter direkt aus Umgebungsvariablen. Spieleserver auf Amazon GameLift Servers Anywhere-Flotten müssen `InitSDK()` mit Serverparametern aufgerufen werden.
 - [C++, C#, Unreal] Bei Server-SDK-Aufrufen wurden die Fehlermeldungen verbessert.

- [C++ SDK] Um die Build-Zeiten für das Server-SDK zu verkürzen, `-DRUN_CLANG_FORMAT` ist das Build-Flag standardmäßig deaktiviert. Sie können es mit aktivieren - `DRUN_CLANG_FORMAT=1`.
- [C++ SDK] Beim Erstellen der Bibliotheken ohne die Standardbibliotheken (`-DGAMELIFT_USE_STD=0`) werden `InitSDK()` keine `std::` Datentypen mehr verwendet.
- Erweiterte Dokumentation zum Server-SDK 5.x
 - Aktualisierte Server-SDK-Referenzhandbücher für C++, C#/Unity und Unreal, einschließlich erweiterter Abdeckung aller Datentypen.
 - [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
 - [C++-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
 - [C++ \(Unreal\) Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)
 - Neue Versionen der Server-SDK 5-Integrationsleitfäden für Unity- und Unreal-Plugins
 - [Amazon GameLift ServersIn ein Unity-Projekt integrieren](#)
 - [Amazon GameLift ServersIn ein Unreal Engine-Projekt integrieren](#)
- Zusätzliche Aktualisierungen der Dokumentation
 - Die Dokumentation für Amazon GameLift Servers Dienst-API-Operationen [GetComputeAccess](#) und [GetInstanceAccess](#) zur Erläuterung der Fernzugriffsverfahren wurde auf der Grundlage der verwendeten Amazon GameLift Servers Server-SDK-Version überarbeitet.
 - Die Beschreibungen wurden überarbeitet [GameSessionPlacement](#), um zu dokumentieren, dass Informationen zur Spielsitzung vorübergehend sind, wenn sich eine Platzierung im Status „ausstehend“ befindet.

13. Juli 2023: Amazon GameLift Servers Fügt Flottenhardwaremetriken hinzu

Sie können jetzt Hardware-Leistungskennzahlen für Ihre Amazon GameLift Servers verwalteten EC2-Flotten verfolgen. Zu den Metriken gehören EC2-Instance-Metriken für CPU-Auslastung, Netzwerkdatenverkehrsvolumen und Festplattenaktivität. read/write Denn Amazon GameLift Servers diese Metriken beschreiben alle aktiven Instances an einem Flottenstandort. Sie können diese Flottenhardwaremetriken mithilfe eines CloudWatch Amazon-Dashboards im anzeigen AWS-Managementkonsole. Sie können sie auch in der Amazon GameLift Servers Konsole unter Flottendetails einsehen.

Weitere Informationen:

- [Überwachen Sie Amazon GameLift Servers mit Amazon CloudWatch](#)(Metriken für Flotten),
Amazon GameLift ServersEntwicklerhandbuch

29. Juni 2023: Amazon GameLift Servers startet Unterstützung für Amazon Linux 2023

Aktualisierte SDK-Versionen: AWS SDK 1.11.111

Amazon GameLift ServersKunden können jetzt das Betriebssystem Amazon Linux 2023 verwenden, um ihre Spielservers zu hosten. AL2023 bietet mehrere Verbesserungen, AL2 unter anderem im Bereich Sicherheit. Dieses Betriebssystem ist in allen Regionen AWS-Regionen mit Ausnahme der China Regionen verfügbar.

Kunden können die neueren Linux-Betriebssysteme verwenden und weiterhin wichtige Sicherheitsupdates erhalten, wenn der Support für Amazon Linux (AL1) im Dezember 2023 endet. Die Support für Amazon Linux 2 läuft bis zum 30. Juni 2026.

Weitere Informationen:

- [Amazon GameLift ServersHäufig gestellte Fragen zum Linux-Server](#)
- [Vergleich von Amazon Linux 2 und Amazon Linux 2023](#)
- Amazon GameLift ServersAPI-Referenzlinks:
 - [AWS SDK-Aktion CreateBuild](#)
 - [CLI-Befehl upload-build](#)
 - [CLI-Befehl create-build](#)

25. Mai 2023: Amazon GameLift Servers FleetIQ Fügt einen Filter hinzu, um Platzierungen von Spielsitzungen auf Instanzen auszuschließen, bei denen das Spiel ausläuft

Aktualisierte SDK-Versionen: SDK 1.11.87 AWS

Wenn du es Amazon GameLift Servers FleetIQ für das Hosten von Spielen verwendest, kannst du jetzt verhindern, dass Spielsitzungen auf Instanzen platziert werden, die derzeit ausgelastet sind. Instances, bei denen das System ausgelastet wird, sind zwar zum Herunterfahren gekennzeichnet, können aber trotzdem ausgewählt werden, um neue Spielsitzungen zu hosten, wenn keine anderen

Hosting-Ressourcen verfügbar sind. Mit dieser neuen Funktion kannst du die Verwendung von Instanzen, die das System belasten, komplett ausschließen.

Nutze diese Funktion, wenn du anrufst `ClaimGameServer`, um nach verfügbaren Spieleservern zu suchen. Fügen Sie den neuen `FilterOption` Parameter hinzu und setzen Sie den Status der erlaubten Instanzen auf Nur AKTIV. Sucht daher nur nach aktiven Instanzen, Amazon GameLift Servers FleetIQ wenn nach einem verfügbaren Spieleserver gesucht und dieser beansprucht wird.

Weitere Informationen:

- [ClaimGameServer](#) in der Amazon GameLift Servers-API-Referenz
- [Wie FleetIQ funktioniert](#) das im Amazon GameLift Servers FleetIQ Entwicklerhandbuch

16. Mai 2023: Amazon GameLift Servers unterstützt die Kennzeichnung der Kostenzuweisung für Flotten

Amazon GameLift Servers Kunden können jetzt Tags zur AWS Billing Kostenzuweisung verwenden, um ihre Hosting-Kosten für Spiele zu organisieren. Sie können einzelnen Amazon GameLift Servers EC2-Flottenressourcen Kostenzuweisungs-Tags zuweisen, um zu verfolgen, wie Ihre Flotten zu den gesamten Hosting-Kosten beitragen.

Weitere Informationen:

- [Tools für Ressourcenkosten und -nutzung](#)
- [Verwendung von AWS Kostenzuweisungs-Tags](#), Benutzerhandbuch AWS Billing

20. April 2023: Amazon GameLift Servers startet die Unterstützung für Windows Server 2016

Aktualisierte SDK-Versionen: AWS SDK 1.11.63

Amazon GameLift Servers Kunden können jetzt das Betriebssystem Windows Server 2016 verwenden, um ihre Spieleserver zu hosten. Dieses Betriebssystem ist in allen Versionen verfügbar AWS-Regionen. Kunden können das neuere Windows-Betriebssystem verwenden und weiterhin wichtige Sicherheitsupdates erhalten, da Microsoft seinen Support für Windows Server 2012 im Oktober 2023 einstellt.

Ab heute müssen Neukunden, die eine Windows-Laufzeitumgebung benötigen, Windows Server 2016 angeben, wenn sie neue Gameserver-Builds für das Hosting erstellen. Bestandskunden

können weiterhin neue Builds und Flotten mit Windows Server 2012 erstellen, müssen jedoch die Migration mit Windows Server 2016 vor dem Ende des Supports von Microsoft am 10. Oktober 2023 abschließen.

Dieses Update beinhaltet die folgenden Serviceänderungen:

- Wenn Sie einen Gameserver-Build mithilfe von Amazon GameLift Servers SDK- oder CLI-Befehlen erstellen, müssen Sie das Betriebssystem jetzt explizit festlegen. Es gibt keinen Standardwert mehr. Verwenden Sie den Wert, um Ihren Spieleserver auf Windows Server 2016 bereitzustellen `WINDOWS_2016`.
- Wenn Sie mit der Amazon GameLift Servers Konsole einen Gameserver-Build erstellen, müssen Sie ein Betriebssystem aus den verfügbaren Werten auswählen. Wenn Sie bereits Kunde mit aktiven Windows Server 2012-Flotten sind, können Sie entweder `WINDOWS_2012` oder `WINDOWS_2016` wählen.

Weitere Informationen:

- Amazon GameLift Servers API-Referenzlinks:
 - [CLI-Befehl `upload-build`](#)
 - [CLI-Befehl `create-build`](#)
 - [AWS SDK-Aktion `CreateBuild`](#)
- [Amazon GameLift Servers Häufig gestellte Fragen zu Windows 2012](#)

13. April 2023: Amazon GameLift Servers startet das Server-SDK 5.x für Unreal

Aktualisierte SDK-Versionen: Server SDK 5.0.0 für Unreal

Die neueste Version des Amazon GameLift Servers Lightweight-Plugins für Unreal Engine basiert jetzt auf dem Amazon GameLift Servers Server-SDK 5.x. Unter den folgenden Links können Amazon GameLift Servers Sie mit der Integration Ihrer Unreal Engine-Umgebung beginnen.

Weitere Informationen:

- [Amazon GameLift Servers In ein Unreal Engine-Projekt integrieren](#)
- [Füge Amazon GameLift Servers es deinem Spieleserver mit dem Server-SDK hinzu](#)
- [C++-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)

14. März 2023: Amazon GameLift Servers startet ein neues Konsolenerlebnis

Die neue Amazon GameLift Servers Konsole beinhaltet die folgenden Verbesserungen:

- **Verbesserte Navigation** — Der neue Navigationsbereich erleichtert die Navigation zwischen Amazon GameLift Servers Ressourcen.
- **Amazon GameLift Servers Landingpage** — Die neue Landingpage enthält Links zu hilfreicher Dokumentation Amazon GameLift Servers, bietet einen allgemeinen Überblick über und bietet Unterstützung durch Links zur Dokumentation, häufig gestellte Fragen und AWS re:Post.
- **Verbesserte CloudWatch Amazon-Metriken** — Amazon GameLift Servers Metriken sind jetzt sowohl in der Amazon GameLift Servers Konsole als auch in Ihren CloudWatch Dashboards verfügbar. Dieses Update enthält auch neue Messwerte für Leistung, Auslastung und Spielsitzungen.

Weitere Informationen:

- [Verwalte Game-Hosting-Ressourcen mit Amazon GameLift Servers](#)
- [Einen FlexMatch Matchmaker aufbauen](#)

14. Februar 2023: unterstützt Amazon GameLift Servers jetzt serverseitige Verschlüsselung für Amazon SNS SNS-Themen

Die serverseitige Verschlüsselung (SSE) für SNS-Themen verschlüsselt Ihre vertraulichen Daten im Ruhezustand. SSE verwendet AWS Key Management Service (AWS KMS) -Schlüssel, um den Inhalt Ihrer SNS-Themen zu schützen.

Weitere Informationen:

- [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#)
- [FlexMatchMatchmaking-Ereignisse](#)
- [Verschlüsselung im Ruhezustand](#)

9. Februar 2023: Das Amazon GameLift Servers Server-SDK unterstützt .NET 6 mit C #10

Aktualisierte SDK-Versionen: Server-SDK 5.0.0 für .NET 6. Es sind keine SDK-Updates erforderlich.

Wenn Sie die Unity Real-Time Development Platform verwenden, verwenden Sie weiterhin das Amazon GameLift Servers Server-SDK 5.0.0 mit .NET 4.6. Unity unterstützt .NET 6 nicht.

Weitere Informationen:

- Laden Sie unter [Amazon GameLift Servers Erste Schritte](#) die neueste Version des Amazon GameLift Servers Server-SDK herunter
- [C#-Server-SDK 5.x für Amazon GameLift Servers -- Aktionen](#)

31. Januar 2023: Das Amazon GameLift Servers Server-SDK unterstützt die Go-Sprache

Aktualisierte SDK-Versionen: Server-SDK 5.0.0 für Go

Weitere Informationen:

- Laden Sie die neueste Version des Amazon GameLift Servers Server-SDK unter [Amazon GameLift Servers Erste Schritte](#) herunter
- [Go-Server-SDK für Amazon GameLift Servers -- Aktionen](#)

1. Dezember 2022: Amazon GameLift Servers startet Amazon GameLift Servers Anywhere und Amazon GameLift Servers Server SDK 5.0

Aktualisierte SDK-Versionen: AWS SDK 1.10.21, Server-SDK 5.0.0 für C++ und C#

Amazon GameLift Servers Anywhere verwendet keine Gameserver-Ressourcen, um Spieleserver zu hosten. Amazon GameLift Servers Sie können Amazon GameLift Servers Anywhere verwenden, um Ihre eigenen Rechenressourcen in Amazon GameLift Servers verwaltetes EC2-Computing zu integrieren, um Ihre Spieleserver auf mehrere Rechenarten zu verteilen. Sie können Amazon GameLift Servers Anywhere auch verwenden, um Ihre Spieleserver iterativ zu testen, ohne den Build Amazon GameLift Servers für jede Iteration hochladen zu müssen.

Höhepunkte:

- Neue Amazon GameLift Servers Anywhere-Flotten- und Compute-Typen
- Amazon GameLift Servers Registrierung von Computing-Ressourcen von Anywhere
- Verbesserter Test-Iterationszyklus

Amazon GameLift ServersDas Server-SDK 5.0.0 bietet Verbesserungen am vorhandenen Server-SDK und einen neuen Ressourcentyp, Compute. Das Server-SDK 5.0.0 unterstützt Amazon GameLift Servers Anywhere und die Verwendung Ihrer eigenen Rechenressourcen für das Hosting von Spieleservern.

Weitere Informationen:

- [Server-SDK 5.x für Amazon GameLift Servers](#)
- [Geografische Standorte](#)
- [Wählen Sie Rechenressourcen für eine verwaltete Flotte](#)
- [Erstelle eine Amazon GameLift Servers Überall Flotte](#)

25. August 2022: Amazon GameLift Servers startet Unterstützung für Local Zones

Aktualisierte SDK-Versionen: AWS SDK 1.9.333

Amazon GameLift Serversist jetzt in acht Local Zones in den Vereinigte Staaten verfügbar, sodass du deine Flotten näher an den Spielern einsetzen kannst. Sie können alle verwalteten Amazon GameLift Servers Funktionen mit Local Zones nutzen, indem Sie die Local Zones zu Ihren Flotten hinzufügen.

Local Zones erweitern AWS Ressourcen und Dienste bis an den Rand der Cloud, in der Nähe von Zentren mit großer Bevölkerung, Industrie und Informationstechnologie (IT). Das bedeutet, dass Sie Anwendungen, die eine Latenz im einstelligen Millisekundenbereich erfordern, näher an Endbenutzern oder lokalen Rechenzentren bereitstellen können.

Weitere Informationen:

- [Amazon GameLift Serverslokale Zonen](#)
- [Geografische Standorte](#)
- [Erstellen Sie eine Amazon GameLift Servers verwaltete EC2 Flotte](#)

28. Juni 2022: Amazon GameLift Servers startet ein neues Opt-In-Konsolenerlebnis

Die neue Amazon GameLift Servers Konsole beinhaltet die folgenden Verbesserungen:

- Verbesserte Navigation — Der neue Navigationsbereich erleichtert die Navigation zwischen Amazon GameLift Servers Ressourcen.

- [Amazon GameLift Servers Landingpage](#) — Die neue Landingpage enthält Links zu hilfreicher Dokumentation Amazon GameLift Servers, bietet einen allgemeinen Überblick über und bietet Unterstützung durch Links zur Dokumentation, häufig gestellte Fragen und AWS re:Post.
- [Verbesserte CloudWatch Amazon-Metriken](#) — Amazon GameLift Servers Metriken sind jetzt sowohl in der Amazon GameLift Servers Konsole als auch in Ihren CloudWatch Dashboards verfügbar. Dieses Update enthält auch neue Messwerte für Leistung, Auslastung und Spielsitzungen.

Weitere Informationen:

- [Verwalte Game-Hosting-Ressourcen mit Amazon GameLift Servers](#)
- [Einen FlexMatch Matchmaker aufbauen](#)

15. Februar 2022: FlexMatch Fügt eine zusammengesetzte Regel und weitere Verbesserungen hinzu

FlexMatchBenutzer haben jetzt Zugriff auf die folgenden Funktionen:

- [Zusammengesetzte Regel](#) — Unterstützung für zusammengesetzte Matchmaking-Regeln für Spiele mit 40 oder weniger Spielern hinzugefügt. Sie können jetzt logische Anweisungen verwenden, um eine zusammengesetzte Regel zu erstellen, um ein Spiel zu bilden. Ohne eine zusammengesetzte Regel in Ihrem Regelsatz müssen alle Regeln im Regelsatz wahr sein, um eine Übereinstimmung zu bilden. Bei zusammengesetzten Regeln können Sie mithilfe der folgenden logischen Operatoren auswählen, welche Regeln angewendet werden sollen: `and`, `or`, `not`, `andxor`.
- [Flexible Teamauswahl](#) — Die Eigenschaftsausdrücke für die Spielerzuweisung wurden aktualisiert und unterstützen nun die Auswahl einer Teilmenge aller verfügbaren Teams.
- [Längere Zeichenkettenlisten](#) — Die maximale Anzahl von Zeichenketten in einer Liste von Zeichenketten mit Spielerattributwerten wurde von 10 auf 100 erhöht.

Weitere Informationen:

- [Amazon GameLift Servers FlexMatch Leitfaden für Entwickler](#)
 - [FlexMatch Regeltypen](#)
 - [FlexMatch Eigenschaftsausdrücke](#)
- [AttributeValue: SL](#)

28. Oktober 2021: Amazon GameLift Servers fügt Unterstützung für Flotten mit mehreren Regionen in der Region Asien-Pazifik (Osaka) Amazon GameLift Servers FleetIQ hinzu; fügt Unterstützung für AWS Graviton2-Prozessoren hinzu

[Aktualisierte SDK-Versionen: SDK 1.9.133 AWS](#)

Amazon GameLift Servers ist jetzt in Region Asien-Pazifik (Osaka) verfügbar. Spieleentwickler können jetzt mithilfe einer Flotte aus GameLift mehreren Regionen Instanzen in Osaka bereitstellen.

Sie können jetzt von Graviton2 gehostete Spieleserver verwenden, die auf der ARM-basierten Prozessorarchitektur basieren, um im Vergleich zu den entsprechenden Intel-basierten Rechenoptionen eine höhere Leistung zu geringeren Kosten zu erzielen.

Höhepunkte:

- Amazon GameLift Servers ist jetzt in Region Asien-Pazifik (Osaka) verfügbar.
- Amazon GameLift Servers FleetIQ Spieleservergruppen können jetzt so konfiguriert werden, dass sie die Graviton2-Instanzfamilien c6g, m6g und r6g verwalten.

Weitere Informationen:

- [Amazon GameLift Servers Flotte mit mehreren Regionen](#)
- [CreateGameServerGroup](#)
- [AWS Graviton-Prozessor](#)

20. September 2021: Amazon GameLift Servers veröffentlicht das Plugin für Unity

Das Amazon GameLift Servers Plugin für Unity Version 1.0.0 enthält Bibliotheken und eine native Benutzeroberfläche, die den Zugriff auf Amazon GameLift Servers Ressourcen und die Integration Amazon GameLift Servers in Ihr Unity-Spiel erleichtert. Sie können das Amazon GameLift Servers Plugin für Unity verwenden, um auf CloudFormation Vorlagen für gängige Spieleszenarien zuzugreifen Amazon GameLift Servers APIs und diese bereitzustellen. Das Plugin enthält auch ein Beispielspiel, das mit den Beispielszenarien funktioniert. Du kannst Amazon GameLift Servers Local verwenden, um Nachrichten zu sehen, die zwischen dem Spielclient und dem Spieleserver ausgetauscht werden, um zu erfahren, wie ein typisches Spiel damit Amazon GameLift Servers interagiert.

Das Plugin für Unity unterstützt Unity 2019.4 LTS und 2020.3 LTS.

Höhepunkte:

- Erstelle ein Beispielspiel mit verschiedenen Szenarien, führe es aus und modifiziere es oder erstelle dein eigenes.
- Stellen Sie AWS CloudFormation Beispielszenarien für typische Spielszenarien bereit, darunter nur Authentifizierung, Flotte mit einer Region, Flotten mit mehreren Regionen mit Warteschlange und benutzerdefiniertem Matchmaker, Spot-Flotten mit Warteschlange und benutzerdefiniertem Matchmaker und FlexMatch

Weitere Informationen:

- [Integration von Spielen mit dem Plugin für Unity Amazon GameLift Servers](#)

30. Juni 2021: FlexMatch fügt die BatchDistance-Regel hinzu

Sie können den Regeltyp BatchDistance verwenden, um eine Zeichenfolge oder ein numerisches Attribut anzugeben, was jedem Segment eine Vielzahl von Vorteilen bietet.

Höhepunkte:

- Bei großen Spielen (>40 Spieler) könnt ihr jetzt dieselbe Balance auf Basis von Fertigkeiten, Modi und Karten erreichen, anstatt die Spieler gleichmäßig nur nach Fähigkeiten auszubalancieren. Stellt sicher, dass sich alle Spieler in einer Fähigkeitsgruppe befinden, gruppiert mehrere numerische Attribute wie Liga oder Spielstil und gruppiert nach Zeichenkettenattributen wie Karte oder Spielmodus. Du kannst im Laufe der Zeit auch Erweiterungen erstellen. Du kannst zum Beispiel eine Erweiterung erstellen, um einem größeren Spielniveau die Teilnahme am Spiel zu ermöglichen, je länger der Spieler wartet.

Für Spiele mit weniger als 40 Spielern können Sie einen neuen vereinfachten Regelausdruck verwenden.

3. Juni 2021: Client-SDK- und Server-SDK-Updates in Amazon GameLift Servers Echtzeit

Aktualisierte SDK-Versionen: Realtime Client SDK 1.2.0, Server SDK 3.4.0 für Unreal

Mit diesem neuesten SDK-Update können Sie IL2 CPP jetzt in Ihre mobilen Anwendungen integrieren, die das RTS Client SDK verwenden, und Best Practices für Frameworks befolgen. Sie

können jetzt auch das Amazon GameLift Servers Server-SDK für Unreal Version 4.26 erstellen. Dieses Update enthält Komponenten, die sich in Ihren Windows- oder Linux-Gameserver integrieren lassen, darunter C++- und C#-Versionen des Amazon GameLift Servers Server-SDK, Amazon GameLift Servers Local und ein Unreal Engine-Plugin.

Höhepunkte:

- Es wurde Unterstützung für IL2 CPP im RTS Client SDK und für die Erstellung der nativen Bibliotheken als Frameworks hinzugefügt, sodass Sie RTS-Clients für die neuesten Mobilgeräte erstellen können.
- Sie können [DescribePlayerSessions\(\)](#) damit Informationen für eine Einzelspieler Sitzung, für alle Sitzungen in einer Sitzung oder für alle Sitzungen abrufen, die mit einer Einzelspieler-ID verknüpft sind.
- Sie können [GetInstanceCertificate\(\)](#) damit den Dateispeicherort eines PEM-codierten TLS-Zertifikats abrufen, das der Flotte und ihren Instances zugeordnet ist.
- Server-SDK-Unterstützung für Unreal Version 4.26 wurde erstellt.
- Das vorhandene C#-SDK, Version 4.0.2, wurde als kompatibel mit Unity 2020.3 verifiziert. Es waren keine SDK-Updates erforderlich.

Weitere Informationen:

- [Amazon GameLift Servers Leitfaden für Entwickler:](#)
 - [DescribePlayerSessions\(\)](#)
 - [GetInstanceCertificate\(\)](#)

23. März 2021: Amazon GameLift Servers Fügt Benachrichtigungen zur Platzierung von Spielsitzungen hinzu

Aktualisierte SDK-Versionen: AWS SDK [1.8.168](#)

Du kannst jetzt Ereignisse verwenden, um die Platzierungsaktivitäten einer Spielsitzung in einer Warteschlange zu überwachen. Erstellen Sie ein Amazon Simple Notification Service (Amazon SNS) -Thema, um Ereignisbenachrichtigungen zu veröffentlichen, oder richten Sie die Ereignisverfolgung mithilfe von CloudWatch Events ein.

Höhepunkte:

- Für jede Warteschlange können Sie eine benutzerdefinierte Textzeichenfolge festlegen, die in allen Ereignisnachrichten enthalten sein soll.
- Wenn Sie ein Amazon SNS Thema verwenden, können Sie zusätzliche Zugriffsbedingungen festlegen, die die Veröffentlichung auf bestimmte Warteschlangen beschränken.

Weitere Informationen:

- Amazon GameLift ServersLeitfaden für Entwickler:
 - [Event-Benachrichtigung für die Platzierung von Spielsitzungen einrichten](#) (neu)
 - [Platzierungsveranstaltungen für Spielsitzungen](#) (neu)
- [API-Referenz \(AWS SDK\)](#)
 - Neue Warteschlangenparameter für Spielsitzungen `NotificationTarget` und `CustomEventData`: [GameSessionQueue](#), [CreateGameSessionQueue](#), [UpdateGameSessionQueue](#)
- [Amazon GameLift Servers-Forum](#)

16. März 2021: Amazon GameLift Servers Fügt Flotten mit mehreren Regionen hinzu, sechs neue Regionen

[Aktualisierte SDK-Versionen: SDK 1.8.163 AWS](#)

Amazon GameLift ServersManaged Hosting ist jetzt in 21 AWS Regionen verfügbar. Die neuen Regionen sind Kapstadt (af-south-1), Bahrain (me-south-1), Hongkong (ap-east-1), Mailand (eu-south-1), Paris (eu-west-3) und Stockholm (eu-north-1).

Mit der neuen Funktion für Flotten mit Amazon GameLift Servers mehreren Standorten kannst du jetzt eine einzige Flotte einrichten, die deine Spieleserver in einer oder allen der 20 Amazon GameLift Servers unterstützten Regionen hostet (außer in der Region Peking). Diese Funktion zielt darauf ab, den Arbeitsaufwand für die Einrichtung und Wartung Amazon GameLift Servers von Hosting-Ressourcen weltweit erheblich zu reduzieren. Flotten mit mehreren Standorten können in den folgenden AWS Regionen eingerichtet werden: us-east-1 (Nord-Virginia), us-west-2 (Oregon), eu-central-1 (Frankfurt), eu-west-1 (Irland), ap-southeast-2 (Sydney), ap-northeast-1 (Tokio) und ap-northeast-2 (Seoul). In allen anderen Regionen können Sie nach Bedarf weiterhin Flotten mit einem Standort einrichten. Alle Flotten, die vor dieser Version erstellt wurden, sind Flotten mit nur einem Standort. Die Verwendung von Flotten mit mehreren Standorten hat keinen Einfluss auf Ihre Hosting-Kosten. Amazon GameLift ServersDie Preisgestaltung richtet sich nach der Art,

dem Standort und dem Volumen der Instances, die Sie verwenden. (Weitere Informationen finden Sie unter [Amazon GameLift ServersPreisgestaltung](#).) CloudFormation Unterstützung für Flotten mit mehreren Standorten wird in Kürze verfügbar sein.

 Note

Flotten mit mehreren Standorten sind in den Regionen China nicht verfügbar. Amazon GameLift Servers Ressourcen, die sich in China Regionen befinden, können nicht mit Ressourcen in anderen Regionen interagieren oder von Ressourcen in anderen Amazon GameLift Servers Regionen genutzt werden.

Höhepunkte:

- Fügen Sie bei einer Flotte mit mehreren Standorten explizit eine Liste mit abgelegenen Standorten hinzu. Amazon GameLift Servers stellt Instances desselben Typs und derselben Konfiguration, einschließlich der Build- und Runtime-Konfiguration, in der Heimatregion der Flotte und an allen hinzugefügten Standorten bereit.
- Passen Sie die Kapazitätseinstellungen und die Skalierung für jeden Standort unabhängig voneinander an. Die Richtlinien für die automatische Skalierung gelten für eine gesamte Flotte, aber Sie können sie je nach Standort ein- oder ausschalten.
- Starte neue Spielsitzungen an bestimmten Flottenstandorten. Wenn du Warteschlangen für Spielsitzungen oder Spielersuche verwendest, um Spielsitzungen zu platzieren, kannst du jetzt nach Standort, Hosting-Kosten und Spielerlatenz priorisieren, wo neue Spielsitzungen beginnen.
- Rufen Sie Hosting-Metriken in der Amazon GameLift Servers Konsole ab, entweder aggregiert für alle Standorte in einer Flotte oder aufgeschlüsselt nach einzelnen Flottenstandorten.

Weitere Informationen:

- [Amazon-Spieletechnik-Blog](#)
- [API-Referenz \(AWS SDK\)](#)
 - Betrieb neuer Flottenstandorte: [CreateFleetLocations](#), [DescribeFleetLocationAttributes](#), [DescribeFleetLocationCapacity](#), [DescribeFleetLocationUtilization](#), [DeleteFleetLocations](#)
 - Aktualisierter Flottenbetrieb mit neuer Unterstützung für mehrere Standorte: [CreateFleet](#), [UpdateFleetCapacity](#), [DescribeEC2InstanceLimits](#), [DescribeInstances](#), [StopFleetActions](#), [StartFleetActions](#)

- Die Platzierung von Spielsitzungen wurde aktualisiert, mit neuer Priorität und Filterfunktion: [CreateGameSessionQueue](#),, [DescribeGameSessionQueuesUpdateGameSessionQueue](#)
- Die Operationen zur Erstellung von Spielsitzungen wurden aktualisiert, mit neuer Unterstützung für den Standort: [CreateGameSession](#), [DescribeGameSessions](#), [DescribeGameSessionDetails](#), [SearchGameSessions](#)
- [Amazon GameLift ServersLeitfaden für Entwickler](#):
 - [Amazon GameLift ServersServicestandorte](#)(aktualisiert)
 - [Anpassungen der Hosting-Ressourcen](#) (neu)
 - [Skalierung der Game-Hosting-Kapazität mit Amazon GameLift Servers](#)(aktualisiert)
 - [Passen Sie eine Warteschlange für Spielsitzungen an](#) (neu)
 - [Flottendetails in der Amazon GameLift Servers Konsole](#)(aktualisiert)
- [Amazon GameLift Servers-Forum](#)

9. Februar 2021: Amazon GameLift Servers erweitert die Unterstützung für AMD-Instanzen, eigenständig FlexMatch

Aktualisierte SDK-Versionen: AWS SDK [1.8.139](#)

Diese Veröffentlichung umfasst folgende Updates:

- Amazon GameLift ServersFleetIQSpieleservergruppen können jetzt so konfiguriert werden, dass sie die AMD-Instanzfamilien C5a, M5a und R5a verwalten. Die unterstützten Amazon EC2 EC2-Instance-Typen, wie sie für aufgeführt sind GameServerGroup [InstanceDefinition](#), umfassen jetzt Folgendes:
 - c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge, c5a.24xlarge
 - m5a.groß, m5a.xgroß, m5a.2xgroß, m5a.4xgroß, m5a.8xgroß, m5a.12xgroß, m5a.16xgroß, m5a.24xgroß
 - r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge

Hinweis: AMD-Instances für FleetIQ sind derzeit nicht für die Verwendung in der AWS Region China (Peking) verfügbar. Weitere Informationen finden Sie unter [Funktionsverfügbarkeit und Implementierungsunterschiede](#) in China.

- Amazon GameLift Servers Managed Game Hosting unterstützt jetzt AMD-Instances in der Region China (Peking), die von Sinnet betrieben werden. Zu den neuen AMD-Instance-Familien gehören M5a und R5a. Zu den unterstützten EC2-Instance-Typen, wie sie für Fleet aufgeführt sind [InstanceType](#), gehören jetzt die folgenden:
 - m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12xlarge, m5a.16xlarge, m5a.24xlarge
 - r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge
- Amazon GameLift Servers FlexMatch kann jetzt als eigenständige Matchmaking-Lösung in der Region China (Peking) verwendet werden, betrieben von Sinnet. Kunden können in der Region Peking einen FlexMatch Matchmaker erstellen und den [FlexMatchMode](#) Parameter auf STANDALONE konfigurieren. Weitere Informationen dazu FlexMatch, entweder mit Amazon GameLift Servers verwaltetem Hosting oder mit einer Amazon GameLift Servers Nicht-Hosting-Lösung, finden Sie im [Amazon GameLift Servers FlexMatch Entwicklerhandbuch](#).
- Beim Einrichten von Ereignisbenachrichtigungen für Amazon GameLift Servers FlexMatch können Sie jetzt ein Amazon SNS SNS-FIFO-Thema als Benachrichtigungsziel festlegen. Weitere Informationen finden Sie unter:
 - [MatchmakingConfiguration NotificationTarget](#), API-Referenz Amazon GameLift Servers
 - [FlexMatch Ereignisbenachrichtigung einrichten](#), Amazon GameLift Servers FlexMatch Entwicklerhandbuch
 - [Wir stellen vor: Amazon SNS FIFO — First-in-first-out Pub/Sub Nachrichten, Nachrichtenblog AWS](#)

22. Dezember 2020: Das Amazon GameLift Servers Server-SDK unterstützt Unreal Engine 4.25 und Unity 2020

Aktualisierte SDK-Versionen: Amazon GameLift Servers Server-SDK 4.0.2, Unreal-Plug-in-Version 3.3.3

Die neueste Version des Amazon GameLift Servers Server-SDK enthält die folgenden Komponenten:

- Das aktualisierte Unreal-Plugin wurde aus Gründen der Kompatibilität mit Unreal Engine 4.25 aktualisiert. Die API wurde nicht geändert.
- Das bestehende C#-SDK, Version 4.0.2, wurde als kompatibel mit Unity 2020 verifiziert. Es waren keine SDK-Updates erforderlich.

Laden Sie unter [Amazon GameLift Servers Erste Schritte](#) die neueste Version des Amazon GameLift Servers Server-SDK herunter.

24. November 2020: Amazon GameLift Servers FlexMatch jetzt für Spiele verfügbar, die überall gehostet werden

Aktualisierte SDK-Versionen: AWS SDK [1.8.95](#)

Amazon GameLift Servers FlexMatch ist ein anpassbarer Matchmaking-Service für Multiplayer-Spiele. Ursprünglich für Nutzer von Amazon GameLift Servers verwaltetem Hosting konzipiert, FlexMatch kann es jetzt in Spiele integriert werden, die andere Hosting-Systeme verwenden peer-to-peer, darunter proprietäres lokales Computing und primitive Cloud-Computing-Typen. Spiele, die Amazon GameLift Servers FleetIQ für das Hosting von Spielen auf Amazon EC2 verwendet werden, können jetzt Matchmaking mit implementieren. FlexMatch

FlexMatch bietet einen robusten Matchmaking-Algorithmus und eine Regelsprache, sodass Sie den Matchmaking-Prozess so anpassen können, dass Spieler anhand der wichtigsten Spielermerkmale und der gemeldeten Latenz zusammengebracht werden. Darüber hinaus FlexMatch bietet es einen Workflow für Matchmaking-Anfragen, der Funktionen wie Spielerpartys, Spielerakzeptanz und Match-Backfill unterstützt. Wenn du FlexMatch mit Amazon GameLift Servers verwaltetem Hosting oder verwendest Amazon GameLift Servers Realtime, verwendet der Matchmaker automatisch, um Hosting-Ressourcen Amazon GameLift Servers zu finden und eine neue Spielsitzung für neu gegründete Spiele zu starten. Bei der Nutzung FlexMatch als eigenständiger Dienst sendet der Matchmaker die Spielergebnisse zurück an Ihr Spiel, das dann mit Ihrer Hosting-Lösung eine neue Spielsitzung starten kann.

API-Operationen für FlexMatch sind Teil der Amazon GameLift Servers Service-API, die im AWS SDK und in der AWS Command Line Interface (AWS CLI) enthalten ist. Diese Version enthält die folgenden Updates zur Unterstützung von eigenständigem Matchmaking:

- Die API-Ressource MatchmakingConfiguration hat die folgenden Änderungen:
 - Neue Eigenschaft, FlexMatchMode gibt an, ob der Matchmaker mit Amazon GameLift Servers verwaltetem Hosting oder als eigenständiges Matchmaking verwendet wird.
 - Die Eigenschaft GameSessionQueueArns ist nicht erforderlich, wenn sie auf FlexMatchMode Standalone gesetzt ist.
 - Diese Eigenschaften werden beim eigenständigen Matchmaking nicht verwendet: AdditionalPlayerCount,, BackfillModeGameProperties,GameSessionData.

- Die automatische Backfill-Funktion ist beim eigenständigen Matchmaking nicht verfügbar.

24. November 2020: AMD-Instanzen sind jetzt verfügbar auf Amazon GameLift Servers

Aktualisierte SDK-Versionen: AWS SDK [1.8.95](#)

Die Liste der Amazon EC2 EC2-Instance-Typen, die Amazon GameLift Servers jetzt unterstützt werden, umfasst drei neue Instance-Familien: C5a, M5a und R5a. Diese Familien bestehen aus rechenoptimierten AMD-Instances, die von AMD EPYC-Prozessoren angetrieben werden, die mit Frequenzen bis zu 3,3 laufen. GHz. Die AMD-Instances sind x86-kompatibel. Spiele, auf denen sie gerade laufen, Amazon GameLift Servers können unverändert für AMD-Instance-Typen bereitgestellt werden. Die neuen Instances sind in den folgenden AWS Regionen verfügbar: USA Ost (Nord-Virginia und Ohio), USA West (Oregon und Nordkalifornien), Zentral-Kanada (Montreal), Südamerika (Sao Paulo), EU-Mitte (Frankfurt), EU West (London und Irland), Asien-Pazifik Süd (Mumbai), Asien-Pazifik-Nordosten (Seoul und Tokio) und Asien-Pazifik Südosten (Singapur und Sydney).

Zu den neuen AMD-Instances gehören:

- c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge, c5a.24xlarge
- m5a.groß, m5a.xgroß, m5a.2xgroß, m5a.4xgroß, m5a.8xgroß, m5a.12xgroß, m5a.16xgroß, m5a.24xgroß
- r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge

Weitere Informationen:

- [Amazon-Spieletechnik-Blog](#)
- [Amazon GameLift ServersPreisgestaltung für Instanzen](#)
- [Amazon EC2 EC2-Instances mit AMD EPYC-Prozessoren](#)
- [Amazon GameLift Servers-Forum](#)

11. November 2020: Versionsupdate für das Server-SDK Amazon GameLift Servers

Aktualisierte SDK-Versionen: Amazon GameLift Servers Server-SDK 4.0.2

Die neue Server-SDK-Version 4.0.2 behebt ein bekanntes Problem mit dem API-Betrieb.

`StartMatchBackfill()` Dieser Vorgang gibt jetzt eine korrekte Antwort auf eine Match-Backfill-Anfrage zurück.

Das Problem hatte keine Auswirkungen auf den Match-Backfill-Prozess, und die Funktionsweise dieser Funktion hat sich nicht geändert. Das Problem hat sich möglicherweise auf die Protokollnachrichten und die Fehlerbehandlung bei Match-Backfill-Anfragen ausgewirkt.

Laden Sie unter [Amazon GameLift Servers Erste Schritte](#) die neueste Version des Amazon GameLift Servers Server-SDK herunter.

5. November 2020: Neue FlexMatch Algorithmusanpassungen

FlexMatchBenutzer können jetzt die folgenden Standardverhaltensweisen für den Matchmaking-Prozess anpassen. Diese Anpassungen sind in einem Matchmaking-Regelsatz festgelegt. Es gibt keine Änderungen an der Amazon GameLift Servers SDKs

- **Backfill-Tickets priorisieren:** Sie können wählen, ob Sie bei der Suche nach akzeptablen Matches die Priorität von Backfill-Tickets erhöhen oder verringern möchten. Die Priorisierung von Backfill-Tickets ist nützlich, wenn die automatische Backfill-Funktion aktiviert ist. Verwenden Sie die Algorithmus-Eigenschaft `backfillPriority`
- **Vorsortierung zur Optimierung der Konsistenz und Effizienz von Spielen:** Konfigurieren Sie Ihren Matchmaker so, dass der Ticketpool vorab sortiert wird, bevor Tickets zur Auswertung gebündelt werden. Indem du Tickets anhand der wichtigsten Spielerattribute vorsortierst, haben deine resultierenden Spiele in der Regel Spieler, die sich in diesen Attributen ähnlicher sind. Sie können den Bewertungsprozess auch effizienter gestalten, indem Sie die Sortierung nach denselben Attributen vornehmen, die auch in den Spielregeln verwendet werden. Verwenden Sie die Algorithmeigenschaft `sortByAttributes`, wenn die `strategy` Eigenschaft auf „sortiert“ gesetzt ist.
- **Passen Sie an, wie die Wartezeiten für Erweiterungen ausgelöst werden:** Wählen Sie, ob Sie Erweiterungen auf der Grundlage des Alters des neuesten (Standard) oder des ältesten Tickets in einem unvollständigen Spiel auslösen möchten. Das Auslösen mit dem ältesten Ticket führt dazu, dass Spiele schneller abgeschlossen werden, während das Auslösen mit dem neuesten Ticket zu einer höheren Spielqualität führt. Verwenden Sie die Algorithmus-Eigenschaft `expansionAgeSelection`

17. September 2020: Amazon GameLift Servers aktualisiert das Server-SDK

Aktualisierte SDK-Versionen: Amazon GameLift Servers Server-SDK 4.0.1

Das neue Server-SDK enthält die folgenden Updates:

- C#-API-Version 4.0.1
 - Der API-Vorgang [TerminateGameSession\(\)](#) wird nicht mehr unterstützt. Ersetze ihn durch einen Aufruf, [ProcessEnding\(\)](#) um sowohl eine Spielsitzung als auch den Serverprozess zu beenden.
 - Ein bekanntes Problem mit dem [GetInstanceCertificate\(\)](#) Vorgang wurde behoben.
 - Die Operation gibt [GetTerminationTime\(\)](#) jetzt einen Wert vom Datentyp zurück `AwsDateTimeOutcome`.
- C++-API-Version 3.4.1
 - Der Vorgang [TerminateGameSession\(\)](#) wird nicht mehr unterstützt. Ersetze ihn durch einen Aufruf, [ProcessEnding\(\)](#) um sowohl eine Spielsitzung als auch den Serverprozess zu beenden.
- Version 3.3.2 des Unreal Engine-Plug-ins
 - Der Vorgang [TerminateGameSession\(\)](#) wird nicht mehr unterstützt. Ersetze ihn durch einen Aufruf, [ProcessEnding\(\)](#) um sowohl eine Spielsitzung als auch den Serverprozess zu beenden.
 - Der Callback-Vorgang `OnUpdateGameSession` wurde hinzugefügt, um Match Backfill [FProcessParameter](#) zu unterstützen.

Laden Sie unter [Amazon GameLift Servers Erste Schritte](#) die neueste Version des Amazon GameLift Servers Server-SDK herunter.

27. August 2020: Amazon GameLift Servers FleetIQ für Spiele-Hosting mit Amazon EC2 (allgemeine Verfügbarkeit)

[Aktualisierte SDK-Versionen: AWS SDK 1.8.36](#)

Die Amazon GameLift Servers FleetIQ Lösung für kostengünstiges, cloudbasiertes Spiele-Hosting auf Amazon EC2 ist jetzt allgemein verfügbar. Amazon GameLift Servers FleetIQ bietet Entwicklern die Möglichkeit, Spielservers direkt auf Amazon EC2-Spot-Instances zu hosten, indem ihre Rentabilität für das Hosting von Spielen optimiert wird. Spieleentwickler können sie Amazon GameLift Servers FleetIQ mit neuen Spielen oder zur Ergänzung der Kapazität vorhandener Spiele nutzen. Diese Lösung unterstützt die Verwendung von Containern oder anderen AWS Diensten wie AWS Shield und Amazon Elastic Container Service (Amazon ECS).

Diese Version für allgemeine Verfügbarkeit enthält die folgenden Updates für die Amazon GameLift Servers FleetIQ Lösung:

- Ein neuer API-Vorgang `DescribeGameServerInstances` gibt Informationen, einschließlich des Status, zu allen aktiven Instanzen einer Amazon GameLift Servers FleetIQ Spieleservergruppe zurück.
- Neue Balancing-Strategie `ON_DEMAND_ONLY`, konfiguriert eine Spieleservergruppe so, dass sie nur On-Demand-Instances verwendet. Sie können die Balancing-Strategie einer Spieleservergruppe jederzeit aktualisieren, sodass Sie bei Bedarf zwischen Spot-Instances und On-Demand-Instances wechseln können.
- Die folgenden Vorschaulemente wurden aus Gründen der allgemeinen Verfügbarkeit entfernt:
 - Verwendung von benutzerdefinierten Sortierschlüsseln für Spieleserverressourcen. Spieleserver können anhand des Zeitstempels der Registrierung sortiert werden.
 - Tagging für Spieleserver-Ressourcen.

16. April 2020: Amazon GameLift Servers aktualisiert das Server-SDK SDK for Unity und Unreal Engine

Aktualisierte SDK-Versionen: Amazon GameLift Servers Server-SDK 4.0.0, Amazon GameLift Servers Local 1.0.5

Die neueste Version des Amazon GameLift Servers Server-SDKs enthält die folgenden aktualisierten Komponenten:

- C# SDK Version 4.0.0 wurde für Unity 2019 aktualisiert.
- Die Unreal-Plug-in-Version 3.3.1 wurde für die Unreal Engine-Versionen 4.22, 4.23 und 4.24 aktualisiert.
- Amazon GameLift ServersDie lokale Version 1.0.5 wurde aktualisiert, um Integrationen zu testen, die das C#-Server-SDK Version 4.0.0 verwenden.

[Laden Sie die neueste Version des Amazon GameLift Servers Server-SDK unter Erste Schritte herunter. Amazon GameLift Servers](#)

2. April 2020: Amazon GameLift Servers FleetIQ verfügbar für das Hosten von Spielen auf EC2 (öffentliche Vorschauversion)

[Aktualisierte SDK-Versionen: AWS SDK 1.7.310](#)

Die Amazon GameLift Servers-FleetIQ-Funktion optimiert die Rentabilität von kostengünstigen Spot-Instances für die Verwendung mit Spiel-Hosting. Diese Funktion wird nun für Kunden erweitert, die ihre Hosting-Ressourcen direkt und nicht über den verwalteten Amazon GameLift Servers-Service verwalten möchten. Diese Lösung unterstützt die Verwendung von Containern oder anderen AWS Diensten wie AWS Shield und Amazon Elastic Container Service (Amazon ECS).

Weitere Informationen:

[GameTech Blogbeitrag](#) auf Amazon GameLift Servers FleetIQ

19. Dezember 2019: Verbessertes AWS Ressourcenmanagement für Amazon GameLift Servers Ressourcen

[Aktualisierte SDK-Versionen: AWS SDK 1.7.249](#)

Sie können jetzt AWS Ressourcenmanagement-Tools mit Amazon GameLift Servers Ressourcen nutzen. Insbesondere allen wichtigen Amazon GameLift Servers Ressourcen — Builds, Skripten, Flotten, Warteschlangen für Spielsitzungen, Matchmaking-Konfigurationen und Matchmaking-Regelsätze — werden jetzt Werte für Amazon Resource Name (ARN) zugewiesen. Ein Ressourcen-ARN bietet eine konsistente Kennung, die in allen AWS Regionen eindeutig ist. Sie können verwendet werden, um ressourcenspezifische AWS Identity and Access Management (IAM) Berechtigungsrichtlinien zu erstellen. Ressourcen wird jetzt ein ARN und auch die bereits vorhandene Ressourcen-ID zugewiesen, die nicht regionsspezifisch ist.

Darüber hinaus unterstützen Amazon GameLift Servers-Ressourcen jetzt Tagging. Sie können Tags verwenden, um Ressourcen zu organisieren, IAM-Berechtigungsrichtlinien zu erstellen, um den Zugriff auf Ressourcengruppen zu verwalten, AWS Kostenaufschlüsselungen anzupassen usw. Verwenden Sie beim Verwalten von Tags für Amazon GameLift Servers-Ressourcen die Amazon GameLift Servers-API-Aktionen `TagResource()`, `UntagResource()` und `ListTagsForResource()`.

Weitere Informationen:

- [TagResource](#) in der Amazon GameLift Servers-API-Referenz
- [Tagging von AWS Ressourcen](#) in der AWS Allgemeinen Referenz

- [Amazon-Ressourcennamen](#) in der AWS allgemeinen Referenz

14. November 2019: Neue CloudFormation Vorlagen, Updates in der Region China (Peking)

Aktualisierte SDK-Versionen: AWS SDK [1.7.210](#)

CloudFormation Vorlagen für Amazon GameLift Servers

Amazon GameLift ServersRessourcen können jetzt über erstellt und verwaltet werden CloudFormation. Die vorhandenen CloudFormation Build- und Flottenvorlagen wurden aktualisiert, um sie an die aktuellen Ressourcen anzupassen. Neue Vorlagen sind jetzt für Skripte, Warteschlangen, Matchmaking-Konfigurationen und Matchmaking-Regelsätze verfügbar. CloudFormation Vorlagen vereinfachen die Verwaltung von Gruppen verwandter AWS Ressourcen erheblich, insbesondere bei der Bereitstellung von Spielen in mehreren Regionen.

Weitere Informationen:

- [Amazon GameLift ServersReferenz zum Ressourcentyp](#) im AWS CloudFormation Benutzerhandbuch
- [Amazon GameLift ServersHosting-Ressourcen verwalten mit CloudFormation](#) im Amazon GameLift ServersEntwicklerhandbuch

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.