



使用者指南

Amazon EC2 Auto Scaling



Amazon EC2 Auto Scaling: 使用者指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任從何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

Amazon EC2 Auto Scaling 是什麼？	1
Amazon EC2 Auto Scaling 的功能	1
Amazon EC2 Auto Scaling 的價格	3
開始使用	3
使用 Auto Scaling 群組	3
Auto Scaling 優點	4
範例：涵蓋變數需求	5
範例：Web 應用程式架構	6
範例：跨可用區域分配執行個體	7
執行個體生命週期	10
擴展	10
服務中的執行個體	11
縮減	11
分開執行個體	12
連接執行個體	12
lifecycle hook	12
進入和結束待命	13
Amazon EC2 Auto Scaling 配額	13
Amazon EC2 Auto Scaling API 的請求限流	15
EC2 終止率	15
其他服務	15
設定	16
準備使用 AWS CLI	16
開始使用	17
教學課程：建立您的第一個 Auto Scaling 群組	17
準備演練	18
步驟 1：建立啟動範本	18
步驟 2：建立單一執行個體 Auto Scaling 群組	19
步驟 3：驗證 Auto Scaling 群組	20
步驟 4：終止 Auto Scaling 群組中的執行個體	21
步驟 5：後續步驟	22
步驟 6：清除	22
教學課程：設定擴展和負載平衡應用程式	23
先決條件	25

步驟 1：設定啟動範本或啟動組態	25
步驟 2：建立 Auto Scaling 群組	28
步驟 3：驗證負載平衡器是否已連接	30
步驟 4：後續步驟	30
步驟 5：清除	31
相關資源	32
啟動範本	33
使用啟動範本的許可	34
啟動範本支援的 API 操作	34
為 Auto Scaling 群組建立啟動範本	34
建立啟動範本 (主控台)	35
變更預設網路介面設定 (主控台)	37
修改儲存組態 (主控台)	39
從現有執行個體建立啟動範本 (主控台)	41
相關資源	41
限制	41
使用進階設定建立啟動範本	42
必要設定	42
進階設定	42
請求 Spot 執行個體	46
Capacity Blocks 適用於 ML 的	47
遷移 Auto Scaling 群組以啟動範本	52
步驟 1：尋找使用啟動組態的 Auto Scaling 群組	52
步驟 2：將啟動組態複製到啟動範本	55
步驟 3：更新 Auto Scaling 群組以使用啟動範本	56
步驟 4：取代您的執行個體	56
其他資訊	57
將 CloudFormation 堆疊遷移至啟動範本	57
尋找使用啟動組態的 Auto Scaling 群組	58
更新堆疊以使用啟動範本	58
了解更新堆疊資源的行為	62
追蹤遷移	62
啟動組態對應參考	63
AWS CLI 使用啟動範本的範例	64
範例使用方式	65
建立基本的啟動範本	65

指定在啟動時標記執行個體的標籤	66
指定要傳遞給執行個體的 IAM 角色	67
指派公有 IP 地址	67
指定啟動時設定執行個體的使用者資料指令碼	67
指定區塊型設備映射	68
指定專用主機以從外部廠商取得軟體授權	68
指定現有網路介面	68
建立多個網路介面	69
管理啟動範本	70
更新 Auto Scaling 群組以使用啟動範本	72
使用 Systems Manager 參數而非 AMI IDs	73
建立指定 AMI 參數的啟動範本	73
確認啟動範本取得正確的 AMI ID	78
相關資源	78
限制	79
啟動組態	80
建立啟動組態	81
建立啟動組態	82
設定 IMDS	84
使用 EC2 執行個體建立啟動組態	86
變更啟動組態	90
Auto Scaling 群組	92
使用啟動範本建立 Auto Scaling 群組	93
使用啟動範本建立群組	94
使用 EC2 啟動精靈建立群組	96
使用多種執行個體類型與購買選項	100
使用啟動組態建立 Auto Scaling 群組	143
使用啟動組態建立群組	144
使用 從執行個體建立群組 AWS CLI	146
同步佈建	151
同步佈建和非同步擴展	152
限制及考量	152
使用同步佈建啟動執行個體	153
更新 Auto Scaling 群組	156
更新 Auto Scaling 執行個體	158
Auto Scaling 群組配置策略和容量變更	159

標記 群組和執行個體	159
標籤命名和使用限制	160
EC2 執行個體標記生命週期	160
標記 Auto Scaling 群組	161
刪除標籤	164
安全標籤	165
控制對標籤的存取	166
使用標籤篩選 Auto Scaling 群組	167
執行個體維護政策	170
概觀	171
在群組上設定執行個體維護政策	176
lifecycle hook	180
lifecycle hook 可用性	181
考量和限制	181
相關資源	183
Auto Scaling 群組中的生命週期關聯運作方式	183
準備新增 lifecycle hook	186
使用執行個體生命週期政策控制執行個體保留	194
擷取目標生命週期狀態	199
將生命週期掛鉤新增至 Auto Scaling 群組	201
在 Auto Scaling 群組中完成生命週期動作	204
教學課程：使用執行個體中繼資料擷取生命週期狀態	205
教學課程：設定叫用 Lambda 函數的 lifecycle hook	213
暖集區	222
核心概念	223
先決條件	225
更新暖集區中的執行個體	226
相關資源	226
限制	226
使用 lifecycle hook	227
為 Auto Scaling 群組建立暖集區	231
檢視運作狀態檢查狀態	233
AWS CLI 使用暖集區的範例	236
Auto Scaling 群組區域轉移	239
Auto Scaling 群組區域轉移概念	239
Auto Scaling 群組的區域轉移運作方式	240

使用區域轉移的最佳實務	241
使用 AWS 管理主控台 或 啟用區域轉移 AWS CLI	242
可用區域分佈	245
分離連接執行個體	245
分離執行個體的考量事項	246
連接執行個體的考量事項	246
使用分離和連接將執行個體移至不同的群組	247
暫時移除執行個體	252
待命狀態的運作方式	252
考量事項	253
待命狀態中的執行個體運作狀態	253
將執行個體設定為待命，以暫時移除執行個體	252
刪除 Auto Scaling 基礎設施	258
刪除 Auto Scaling 群組	258
(選用) 刪除啟動組態	259
(選用) 刪除啟動範本	259
(選用) 刪除負載平衡器和目標群組	260
(選用) 刪除 CloudWatch 警示	261
資源刪除保護	262
取代您的執行個體	268
執行個體重新整理	268
執行個體重新整理的運作方式	269
了解預設值	274
啟動執行個體重新整理	277
監控執行個體重新整理	286
取代根磁碟區	289
取消執行個體重新整理	293
使用復原來復原變更	294
使用略過相符項目	298
新增檢查點	307
執行個體生命週期上限	312
考量事項	312
設定執行個體生命週期上限	313
限制	314
擴展群組	315
選擇您的擴展方法	315

設定擴展限制	316
設定預設執行個體暖機期	318
擴展效能考量	319
選擇預設執行個體暖機時間	319
為群組啟用預設執行個體暖機期	320
驗證群組的預設執行個體暖機期	321
尋找具有先前設定之執行個體暖機時間的擴展政策	322
清除先前為擴展政策設定的執行個體暖機期	323
手動擴展	324
更改您的 Auto Scaling 群組的所需容量	324
終止 Auto Scaling 群組中的執行個體 (AWS CLI)	327
排程擴展	328
排程擴展的運作方式	329
週期性排程	330
時區	330
考量事項	331
限制	331
建立排程動作	331
檢視排程動作詳細資訊	333
刪除排程動作	334
動態擴展	335
動態擴展政策的運作方式	336
多個動態擴展政策	337
目標追蹤擴展政策	338
步進和簡易擴展政策	353
擴展冷卻時間	367
以 Amazon SQS 為基礎的擴展政策	369
驗證擴展活動	375
停用擴展政策	377
刪除 Auto Scaling 群組的擴展政策	380
AWS CLI 擴展政策的範例	382
預測擴展	385
預測擴展的運作方式	386
建立預測性擴展政策	389
評估您的預測擴展政策	396
覆寫預測	404

使用自訂指標	409
控制執行個體終止	418
終止政策情境	419
設定終止政策	422
使用 Lambda 建立自訂終止政策	427
執行個體縮減保護	433
為正常終止執行個體而設計	437
暫停 - 繼續處理程序	443
程序類型	444
考量事項	444
暫停程序	445
繼續程序	446
暫停程序如何影響其他程序	447
監控	450
運作狀態檢查	451
關於運作狀態檢查	453
設定運作狀態檢查寬限期	458
監控受損的 Amazon EBS 磁碟區	460
設定自訂運作狀態檢查	463
檢視運作狀態檢查失敗的原因	465
故障診斷運作狀態不佳的執行個體	466
使用 監控 Health 儀板表	469
監控 CloudWatch 指標	470
在 Amazon EC2 Auto Scaling 主控台中檢視監控圖表	471
Amazon EC2 Auto Scaling 的 CloudWatch 指標	474
設定 Auto Scaling 執行個體的監控	481
使用 CloudTrail 記錄 API 呼叫	483
CloudTrail 中的 Auto Scaling 管理事件	484
Auto Scaling 事件範例	484
CloudWatch 上的 Auto Scaling RemoveAction 呼叫	485
Amazon SNS 通知選項	485
Amazon SNS 和 Amazon EC2 Auto Scaling	486
使用其他服務	492
容量重新平衡	492
概要	493
容量重新平衡行為	493

考量事項	494
啟用容量重新平衡	496
Capacity Reservations	501
容量保留偏好設定	501
可用區域餘額和容量保留	503
使用容量保留偏好設定	503
可中斷的容量保留	505
AWS CloudShell	508
CloudFormation	509
Amazon EC2 Auto Scaling 和 CloudFormation 範本	509
進一步了解 CloudFormation	510
Compute Optimizer	510
限制	511
調查結果	511
檢視建議	511
評估建議的考量	512
Elastic Load Balancing	513
Elastic Load Balancing 類型	514
準備連接負載平衡器	515
連接負載平衡器	517
設定負載平衡器	520
確認連接狀態	521
新增可用區域	522
移除可用區域	524
分開負載平衡器	519
AWS CLI 使用 Elastic Load Balancing 的範例	526
VPC Lattice	533
準備連接目標群組	535
連接 VPC Lattice 目標群組	537
確認連接狀態	542
EventBridge	542
Amazon EC2 Auto Scaling 事件參考	543
執行個體重新整理範例事件和模式	549
暖集區範例事件和模式	556
EventBridge 規則	561
Amazon VPC	566

預設 VPC	567
非預設 VPC	567
選擇 VPC 子網路時的注意事項	567
VPC 中的 IP 地址指定	568
VPC 中的網路介面	568
執行個體安置租用	569
AWS Outposts	569
了解 VPC 的更多資源	569
安全	571
基礎設施安全性	571
相關資源	572
恢復能力	572
相關資源	573
資料保護	573
使用 AWS KMS keys 加密 Amazon EBS 磁碟區	574
相關資源	575
AWS KMS 用於加密磁碟區的金鑰政策	575
身分和存取權管理	580
存取控制	581
Amazon EC2 Auto Scaling 與 IAM 搭配運作的方式	581
API 許可	589
受管政策	590
服務連結角色	594
身分型政策範例	599
預防跨服務混淆代理人	608
在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量	610
在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色	617
法規遵循驗證	620
PCI DSS 合規	620
使用 VPC 端點進行私有連線	621
建立介面 VPC 端點	621
建立 VPC 端點政策	621
使用 AWS SDKs	623
程式碼範例	625
基本概念	627
Hello Auto Scaling	629

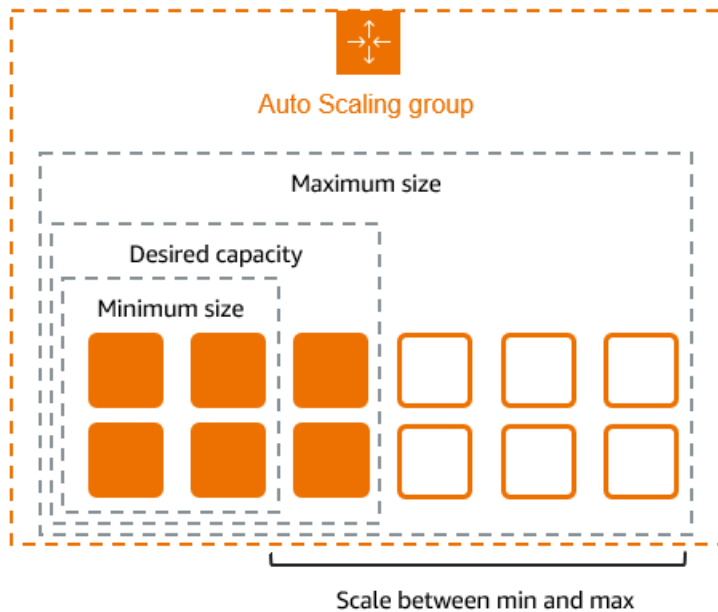
了解基本概念	639
動作	741
案例	962
建置及管理彈性服務	962
疑難排解	1131
擷取錯誤訊息	1131
關閉擴展活動	1133
其他疑難排解資源	1134
執行個體啟動失敗	1134
目前並不支援要求的組態。	1135
安全群組 (安全群組的名稱) 不存在。啟動 EC2 執行個體失敗。	1136
金鑰對 (與您的 EC2 執行個體關聯的金鑰對) 不存在。啟動 EC2 執行個體失敗。	1136
您請求的執行個體類型 (<執行個體類型>) 在您請求的可用區域 (執行個體可用區域) 並不受支援... ..	1136
您的 Spot 請求價格 0.015 低於 Spot 請求所需的最低履行價格 0.0735... ..	1137
裝置名稱 <device name> 無效/裝置名稱上傳無效。啟動 EC2 執行個體失敗。	1137
適用於參數 virtualName 的值 (與執行個體儲存體裝置關聯的名稱) 無效... 啟動 EC2 執行個體失敗。	1138
執行個體存放區 AMI 不支援 EBS 區塊型設備映射。	1138
置放群組可能無法與「<執行個體類型>」搭配使用。啟動 EC2 執行個體失敗。	1138
Client.InternalError：啟動時的用戶端錯誤。	1139
我們目前在您要求的可用區域中，並沒有足夠的 (執行個體類型) 容量... 啟動 EC2 執行個體失敗。	1140
請求的保留沒有足夠的相容性和可用容量來滿足此要求。啟動 EC2 執行個體失敗。	1140
您的容量區塊保留 <保留 id> 尚未啟動。啟動 EC2 執行個體失敗。	1141
沒有符合請求的 Spot 容量。啟動 EC2 執行個體失敗。	1141
(執行個體數) 個執行個體已在運作中。啟動 EC2 執行個體失敗。	1141
AMI 問題	1141
AMI ID (您的 AMI 的 ID) 不存在。啟動 EC2 執行個體失敗。	1142
AMI (AMI ID) 等待處理中，無法執行。啟動 EC2 執行個體失敗。	1142
無效的裝置名稱 <裝置名稱>。啟動 EC2 執行個體失敗。	1142
指定執行個體類型的架構「arm64」與指定 AMI 的架構「x86_64」不相符...啟動 EC2 執行個體失敗。	1143
AMI「<AMI ID>」已停用，無法執行。啟動 EC2 執行個體失敗。	1144
負載平衡器問題	1144
未找到一個或多個目標群組。驗證負載平衡器組態失敗。	1145

找不到負載平衡器 <您的負載平衡器>。驗證負載平衡器組態失敗。	1145
沒有名為 (負載平衡器名稱) 的作用中負載平衡器。更新負載平衡器組態失敗。	1146
EC2 執行個體 (執行個體 ID) 不在 VPC 中。更新負載平衡器組態失敗。	1146
啟動範本問題	1146
您必須使用有效的完整格式啟動範本 (無效值)	1146
您沒有使用啟動範本的授權 (許可不足)	1147
相關資訊	1149
文件歷史紀錄	1151
.....	mclxxx

Amazon EC2 Auto Scaling 是什麼？

Amazon EC2 Auto Scaling 能確保您有數量正確的 Amazon EC2 執行個體來處理應用程式的負載。您可以建立 EC2 執行個體的集合，此集合稱為「Auto Scaling 群組」。您可以為每一個 Auto Scaling 群組指定執行個體數量的最小值，如此 Amazon EC2 Auto Scaling 可確保您的群組數絕不會低於此值。您可以為每一個 Auto Scaling 群組指定執行個體數量的最大值，如此 Amazon EC2 Auto Scaling 可確保您的群組數絕不會大於此值。如果指定了所需要的容量，在建立群組之時或之後的任何時刻，Amazon EC2 Auto Scaling 都會確保您的群組擁有這個數量的執行個體。如果指定了擴展政策，Amazon EC2 Auto Scaling 就會隨著應用程式需求的增加或減少，而啟動或終止執行個體。

例如，下列 Auto Scaling 群組的最小大小為 4 個執行個體、所需容量為 6 個執行個體，最大大小為 12 個執行個體。您定義的擴展政策，會根據您指定的條件，在您的執行個體數最大和最小值間，調整執行個體的數量。



Amazon EC2 Auto Scaling 的功能

使用 Amazon EC2 Auto Scaling，您的 EC2 執行個體會組織成 Auto Scaling 群組，以便將其視為邏輯單位，以進行擴展和管理。Auto Scaling 群組使用啟動範本（或啟動組態）做為其 EC2 執行個體的組態範本。

以下是 Amazon EC2 Auto Scaling 的主要功能：

監控執行中執行個體的運作狀態

Amazon EC2 Auto Scaling 會使用 EC2 運作狀態檢查自動監控執行個體的運作狀態和可用性，並取代已終止或受損的執行個體，以維持所需的容量。

自訂運作狀態檢查

除了內建的運作狀態檢查之外，您還可以定義應用程式特有的自訂運作狀態檢查，以確認其是否如預期回應。如果執行個體未通過您的自訂運作狀態檢查，會自動取代它以維持所需的容量。

在可用區域之間平衡容量

您可以為 Auto Scaling 群組指定多個可用區域，而 Amazon EC2 Auto Scaling 會隨著群組擴展，跨可用區域平均平衡執行個體。這透過保護您的應用程式免於在單一位置發生故障，提供高可用性和彈性。

使用多種執行個體類型與購買選項

在單一 Auto Scaling 群組中，您可以啟動多個執行個體類型和購買選項 (Spot 和隨需執行個體)，讓您透過 Spot 執行個體用量來最佳化成本。您也可以搭配群組中的隨需執行個體使用預留執行個體和 Savings Plans 折扣。

自動替換 Spot 執行個體

如果您的群組包含 Spot 執行個體，Amazon EC2 Auto Scaling 可以在 Spot 執行個體中斷時自動請求替代 Spot 容量。透過容量重新平衡，Amazon EC2 Auto Scaling 也可以監控並主動取代具有較高中斷風險的 Spot 執行個體。

Load balancing

您可以使用 Elastic Load Balancing 負載平衡和運作狀態檢查，確保應用程式流量均勻分佈至運作狀態良好的執行個體。每當執行個體啟動或終止時，Amazon EC2 Auto Scaling 會自動從負載平衡器註冊和取消註冊執行個體。

可擴展性

Amazon EC2 Auto Scaling 也提供數種擴展 Auto Scaling 群組的方式。使用自動擴展可讓您透過新增處理尖峰負載的容量，以及在需求較低時移除容量，來維持應用程式可用性並降低成本。您也可以視需要手動調整 Auto Scaling 群組的大小。

執行個體重新整理

當您更新 AMI 或啟動範本時，執行個體重新整理功能提供以滾動方式更新執行個體的機制。您也可以使用稱為 Canary 部署的分階段方法，在小型執行個體上測試新的 AMI 或啟動範本，然後再將其推展到整個群組。

lifecycle hook

生命週期掛鉤有助於定義在新執行個體啟動時或在執行個體終止之前叫用的自訂動作。此功能特別適用於建置事件驅動型架構，但也可協助您管理執行個體的整個生命週期。

支援具狀態工作負載

生命週期掛鉤也提供在關閉時持續狀態的機制。為了確保具狀態應用程式的持續性，您也可以使用縮減保護或自訂終止政策，以防止具有長時間執行程度的執行個體提早終止。

如需有關 Amazon EC2 Auto Scaling 優點的詳細資訊，請參閱[應用程式架構的 Auto Scaling 優勢](#)。

Amazon EC2 Auto Scaling 的價格

Amazon EC2 Auto Scaling 無需額外費用，因此您可以輕鬆地試用它，並查看它如何讓您的 AWS 架構受益。您只需支付使用 AWS 的資源（例如 EC2 執行個體、EBS 磁碟區和 CloudWatch 警示）。

開始使用

若要開始，請完成[建立您的第一個 Auto Scaling 群組](#)教學課程，以建立 Auto Scaling 群組，並查看該群組中的執行個體終止時的回應方式。

使用 Auto Scaling 群組

您可以使用下列任一介面來建立、存取和管理您的 Auto Scaling 群組：

- AWS 管理主控台 – 提供 Web 介面，讓您用來存取 Auto Scaling 群組。如果您已註冊 AWS 帳戶，則可以登入來存取 Auto Scaling 群組 AWS 管理主控台，使用導覽列上的搜尋方塊來搜尋 Auto Scaling 群組，然後選擇 Auto Scaling 群組。
- AWS Command Line Interface (AWS CLI) – 提供一組廣泛的命令 AWS 服務，並支援 Windows、macOS 和 Linux。若要開始使用，請參閱[準備使用 AWS CLI](#)。如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [autoscaling](#)。
- AWS Tools for Windows PowerShell – 為在 PowerShell 環境中編寫指令碼的人員提供廣泛的 AWS 產品命令。若要開始使用，請參閱《[AWS Tools for PowerShell 使用者指南](#)》。如需詳細資訊，請參閱《[AWS Tools for PowerShell Cmdlet 參考](#)》。
- AWS SDKs – 提供特定語言的 API 操作，並負責許多連線詳細資訊，例如計算簽章、處理請求重試和處理錯誤。如需詳細資訊，請參閱 [AWS 開發套件](#)。

- Query API – 提供您可以使用 HTTPS 請求呼叫的低層級 API 動作。使用查詢 API 是存取 AWS 服務最直接的方式。不過，查詢 API 需要您的應用程式處理低階詳細資訊，例如產生雜湊以簽署要求以及處理錯誤。如需詳細資訊，請參閱《[Amazon EC2 Auto Scaling API 參考](#)》。
- CloudFormation – 支援使用 CloudFormation 範本建立 Auto Scaling 群組。如需詳細資訊，請參閱[使用 建立 Auto Scaling 群組 AWS CloudFormation](#)。

若要以程式設計方式連線至 AWS 服務，您可以使用端點。如需 Amazon EC2 Auto Scaling 呼叫端點的相關資訊，請參閱《[機密區域使用者指南](#)》中的中國 [Amazon Web Services 端點和 ARN](#)》中的 [Amazon EC2 Auto Scaling 端點和配額](#)。AWS 一般參考《中國 Amazon Web Services 入門》中的「最高機密區域」使用者指南》中的「最高

應用程式架構的 Auto Scaling 優勢

將 Amazon EC2 Auto Scaling 新增至應用程式架構，是將 AWS 雲端的優勢最大化的一種方式。使用 Amazon EC2 Auto Scaling 後，您的應用程式將獲得以下好處：

- 較佳的容錯能力。Amazon EC2 Auto Scaling 可以偵測並終止運作狀態不佳的執行個體，然後啟動執行個體來取代它。您也可設定 Amazon EC2 Auto Scaling，以使用於多個可用區域。如果一個可用區域無法使用，Amazon EC2 Auto Scaling 可以在另一個可用區域啟動執行個體來補償。
- 較佳的可用性。Amazon EC2 Auto Scaling 可協助您確保應用程式隨時都有適當的容量，能夠處理目前的流量需求。
- 較佳的成本管理。Amazon EC2 Auto Scaling 可以視需要動態增加和減少容量。因為您付費使用 EC2 執行個體，需要執行個體時再啟用它們，不需要時則終止它們，這樣可以節省費用。

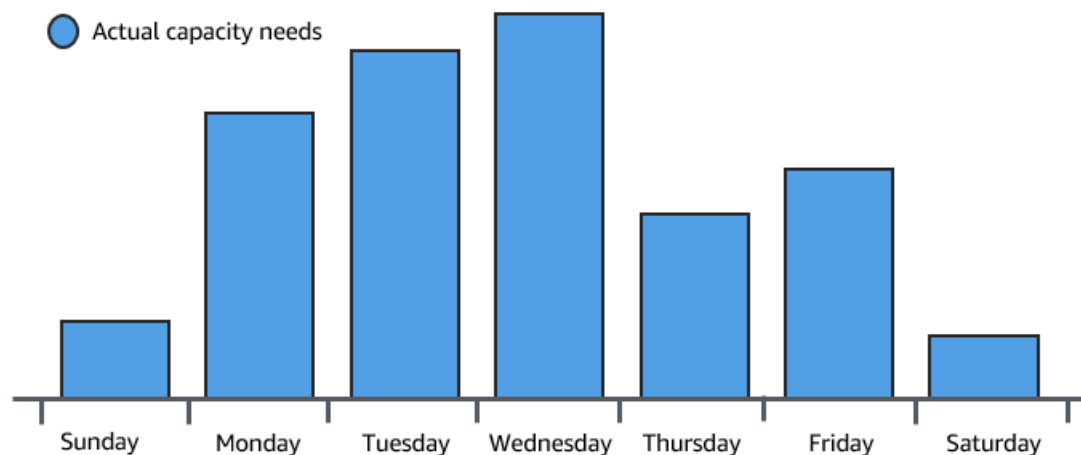
目錄

- [範例：涵蓋變數需求](#)
- [範例：Web 應用程式架構](#)
- [範例：跨可用區域分配執行個體](#)
 - [執行個體分佈](#)
 - [重新平衡活動](#)

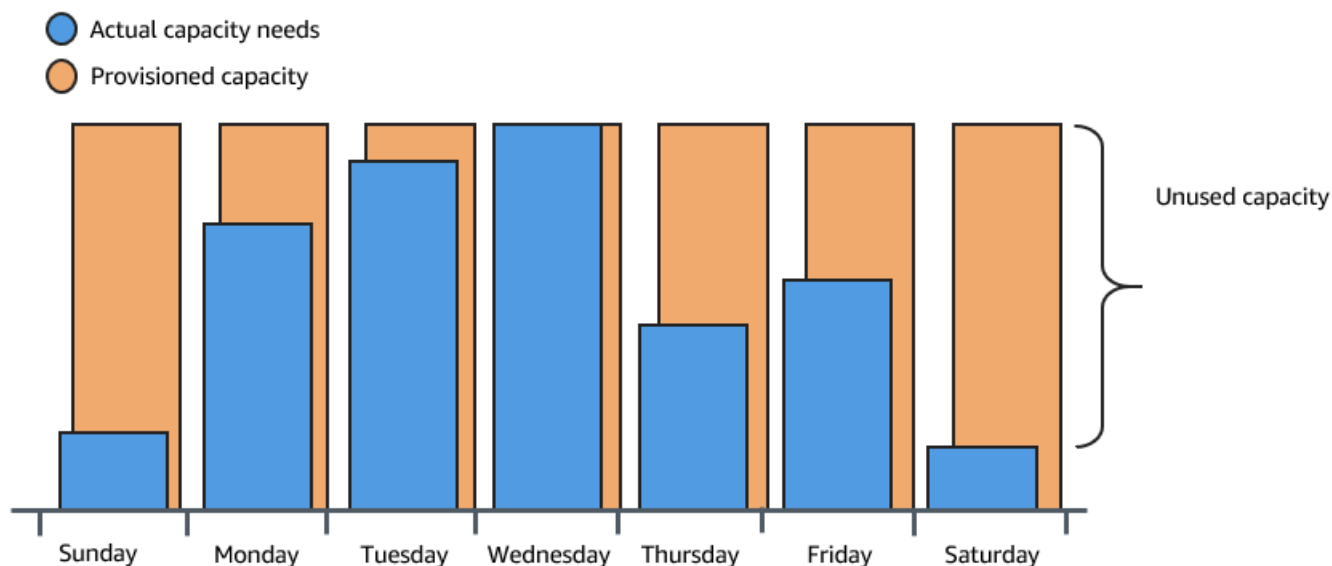
範例：涵蓋變數需求

為了展現 Amazon EC2 Auto Scaling 的一些優點，請考慮在 AWS 上執行基本 Web 應用程式。這個應用程式可讓員工搜尋想要用來開會的會議室。在一週開始和結束時，此應用程式使用最少。在週間時，許多員工會利用此排程會議，因此大幅提高此應用程式的需求。

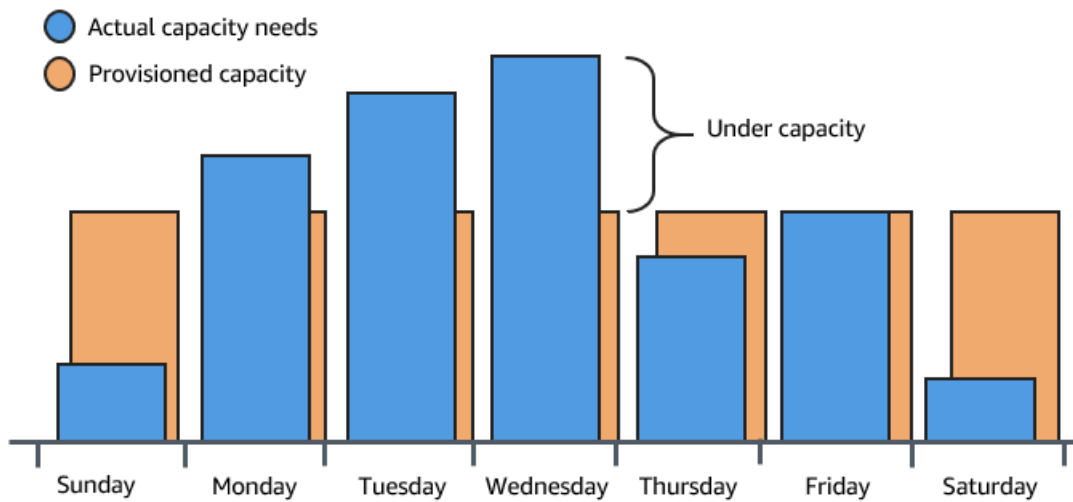
下圖顯示一週間，此應用程式所使用的容量。



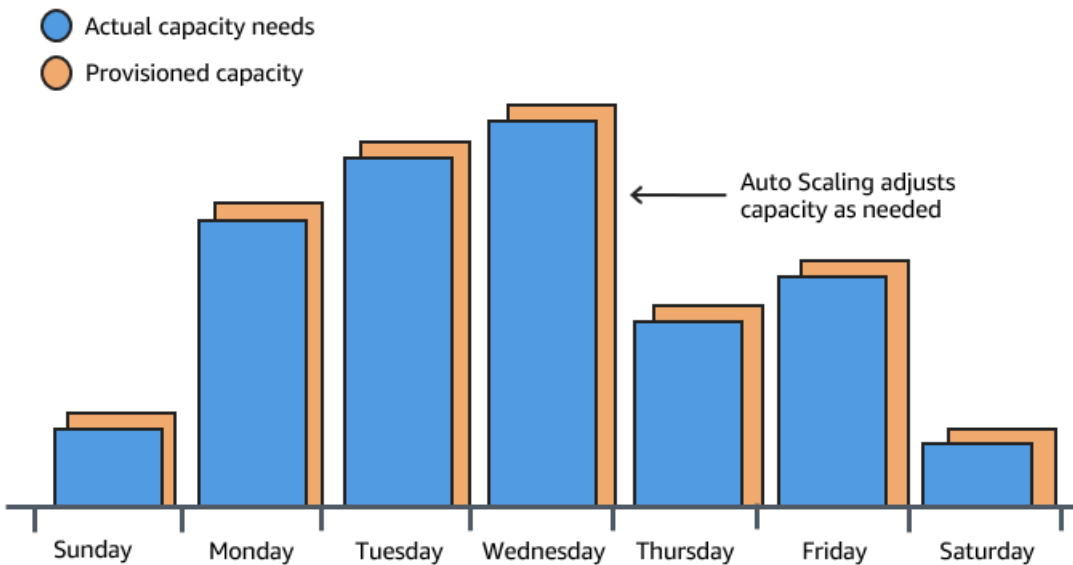
按照傳統做法，有兩種方式可以針對這些容量做變化。第一個選擇是新增足夠的伺服器，讓應用程式永遠有足夠的容量，可以滿足需求。不過，此選項有缺點在於當有天應用程式不再需要這麼多的容量時。未使用的多餘容量，會使維持應用程式執行的成本提高。



第二個選擇是擁有足夠的容量來處理應用程式的平均需求。此選擇較便宜，因為您未購買偶爾使用的設備。不過，當應用程式需求超過其容量時，會有使客戶體驗品質下降的風險。



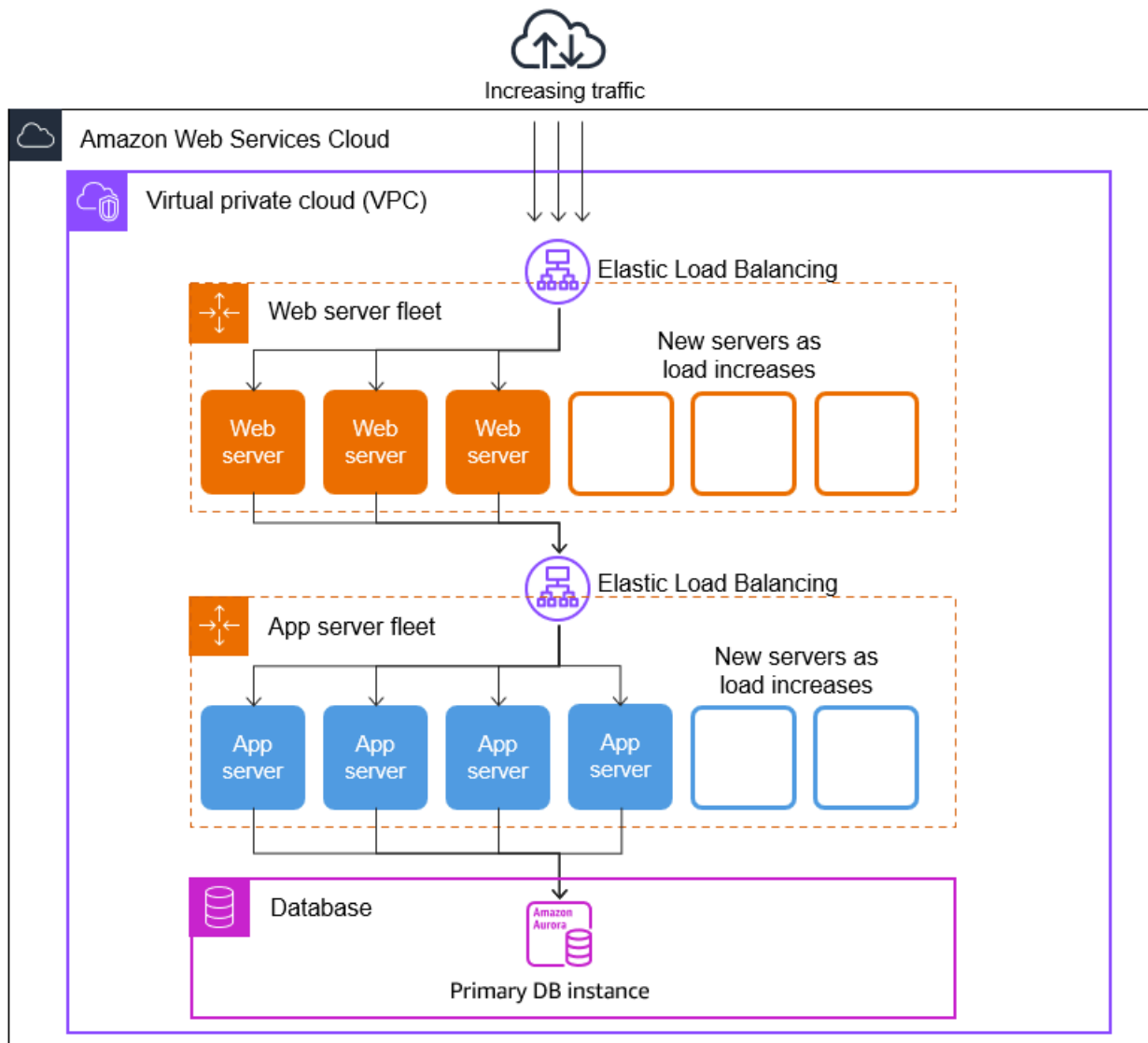
透過新增 Amazon EC2 Auto Scaling 到此應用程式上，您有第三個方案。您可以在需要時新增新的執行個體到應用程式上，且能在不需要它們的時候，將其終止。由於 Amazon EC2 Auto Scaling 使用 EC2 執行個體，當您使用時，您只需支付所使用之執行個體的费用。您現在擁有成本效益的架構，可供客戶體驗最佳的使用品質，同時又能減少費用。



範例：Web 應用程式架構

在常見的 Web 應用程式案例中，您可同時執行多個應用程式的複本，以涵蓋客戶流量的磁碟區。這些應用程式的多個副本由相同的 EC2 執行個體託管（雲端伺服器），每個都會處理客戶的要求。

Amazon EC2 Auto Scaling 會代表您管理 EC2 執行個體的啟動與終止。您可以定義一組條件（例如 Amazon CloudWatch 警示），判斷 Auto Scaling 群組何時啟動或終止 EC2 執行個體。將 Auto Scaling 群組新增到您的網路架構，有助於提高應用程式的可用性和容錯能力。



您可以視需要建立任意數量的 Auto Scaling 群組。例如，您可以為每個層級建立 Auto Scaling 群組。

若要將流量分散到 Auto Scaling 群組中的執行個體間，您可以在架構中引進負載平衡器。如需詳細資訊，請參閱[Elastic Load Balancing](#)。

範例：跨可用區域分配執行個體

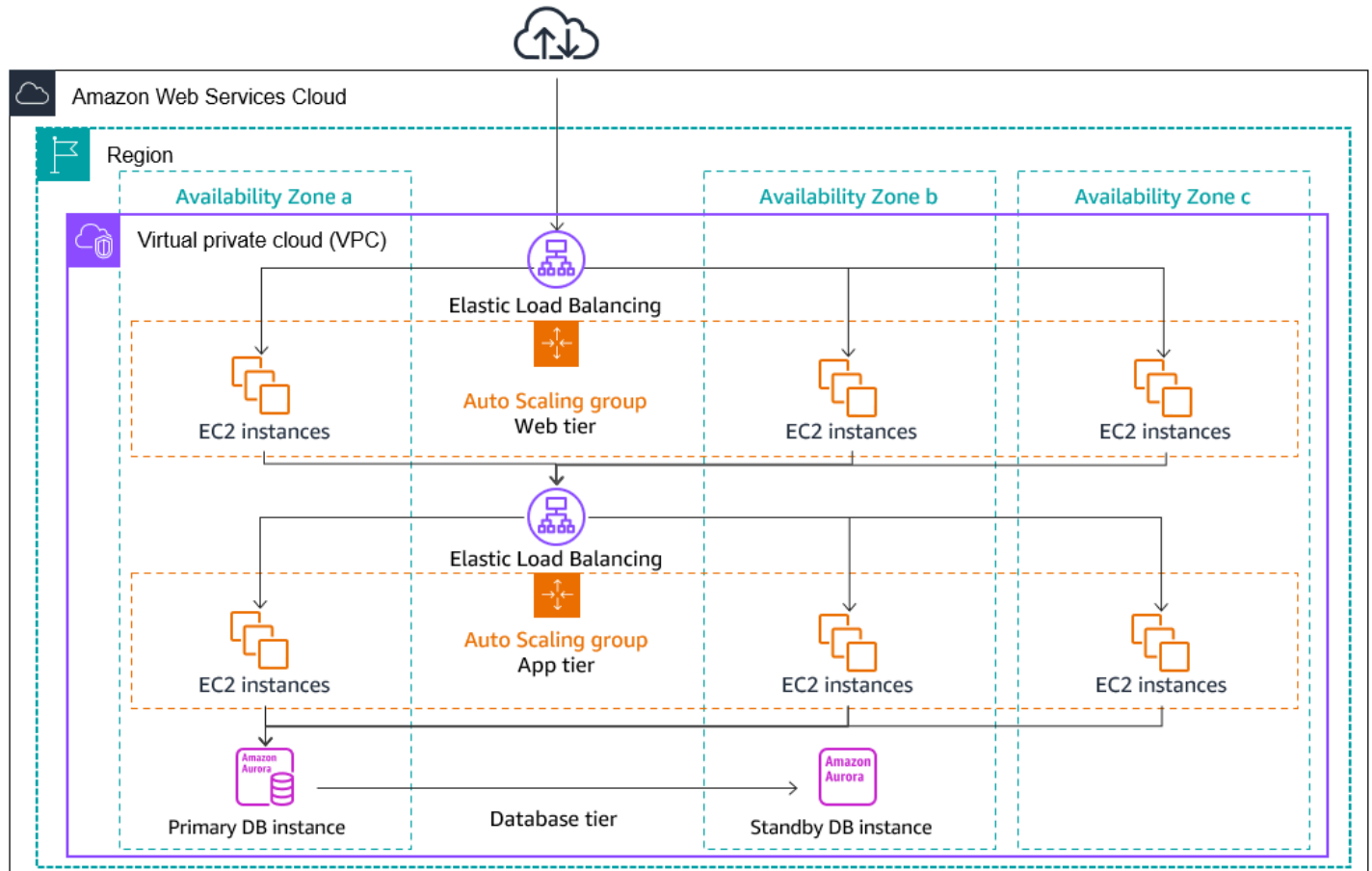
可用區域是給定 AWS 區域內的隔離位置。每個區域都有多個可用區域，旨在為該區域提供高可用性。可用區域是獨立的，因此當您將應用程式設計為使用多個區域時，可提高應用程式的可用性。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 中的彈性](#)。

可用區域由 AWS 區域 程式碼識別，後面接著字母識別符（例如，us-east-1a）。如果選擇建立 VPC 和子網路，而非使用預設 VPC，則可以在每個可用區域中定義一個或多個子網路。各個子網必

須完全位於某一可用區域內，不得跨越多個區域。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 [Amazon VPC 的運作方式](#)。

當您建立 Auto Scaling 群組時，您必須選擇要在其中部署 Auto Scaling 群組的 VPC 和子網路。Amazon EC2 Auto Scaling 會在您選擇的子網路中建立執行個體。因此，每個執行個體都與 Amazon EC2 Auto Scaling 選擇的特定可用區域相關聯。執行個體啟動時，Amazon EC2 Auto Scaling 會嘗試在區域之間平均分配，以實現高可用性和可靠性。

下圖顯示跨三個可用區域部署的多層架構概觀。



執行個體分佈

Amazon EC2 Auto Scaling 會自動嘗試在每個已啟用的可用區域中維持同等數量的執行個體。為此，Amazon EC2 Auto Scaling 會嘗試在可用區域中使用最少的執行個體啟動新的執行個體。如果為可用區域選擇多個子網路，Amazon EC2 Auto Scaling 會嘗試啟動子網路中可用 IP 地址數量最高的執行個體。不過，若嘗試失敗了，Amazon EC2 Auto Scaling 則會在另一個可用區域中嘗試啟動執行個體，直到成功為止。

在可用區域運作狀態不佳或無法使用的情況下，執行個體在各可用區域之間的分佈可能會變得不均勻。當可用區域恢復後，Amazon EC2 Auto Scaling 會自動重新平衡 Auto Scaling 群組。方法為在已啟用且具有最少執行個體的可用區域中啟動執行個體並終止其他地方的執行個體。

重新平衡活動

重新平衡活動分為兩類：可用區域重新平衡和容量重新平衡。

可用區域重新平衡

在特定動作發生後，Auto Scaling 群組會在可用區域間呈現不平衡狀態。Amazon EC2 Auto Scaling 會重新平衡可用區域來加以補償。以下動作可以觸發重新平衡活動：

- 您變更與您 Auto Scaling 群組相關聯的可用區域。
- 您明確終止或分開執行個體，或將執行個體置於待命狀態，然後群組就變得不平衡。
- 先前容量不足的可用區域恢復，現在具有額外的容量。
- 先前可用區域的 Spot 價格高過最高價，現在 Spot 價格則低於最高價格。

重新平衡時，Amazon EC2 Auto Scaling 會先啟動新的執行個體，然後再終止之前的執行個體。以此方式，重新平衡不會影響您應用程式的性能或可用性。

由於 Amazon EC2 Auto Scaling 會在終止之前的執行個體前嘗試啟動新執行個體，在達到或接近最大指定容量時即會阻止或完全停止重新平衡的活動。

若要避免這個問題，在重新平衡活動期間，系統可以暫時超過指定的群組最大容量。預設方法為採用 10% 的餘裕或一個執行個體 (以較大者為準)。只有當群組達到最大容量或接近最大容量且需要重新平衡時，才會擴大餘裕。只要需要重新平衡群組，邊界就會延伸 (通常只會持續幾分鐘)。

或者，您也可以使用執行個體維護政策為 Auto Scaling 群組建立閾值，而群組只能在該閾值範圍內增減容量。這樣，您就可以控制群組重新平衡自身的速度。如需詳細資訊，請參閱[執行個體維護政策](#)。

容量重新平衡

使用 Spot 執行個體時，您可以打開 Auto Scaling 群組的容量重新平衡功能。這讓 Amazon EC2 Auto Scaling 會在 Amazon EC2 報告 Spot 執行個體的中斷風險提高時，嘗試啟動 Spot 執行個體。啟動新的執行個體後，即會終止之前的執行個體。如需詳細資訊，請參閱[Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體](#)。

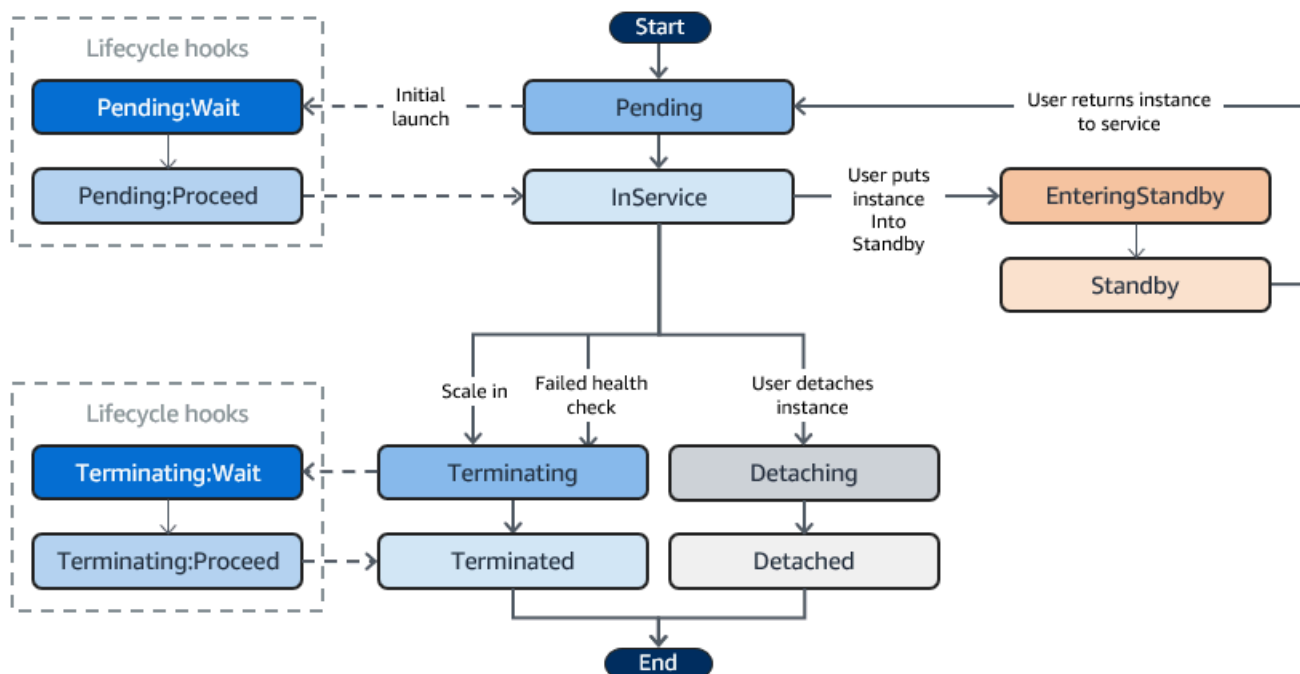
Amazon EC2 Auto Scaling 執行個體生命週期

Auto Scaling 群組中 EC2 執行個體有路徑或者生命週期，這使其與其他 EC2 執行個體有所不同。當 Auto Scaling 群組啟動執行個體並使其執行服務，生命週期就會開始。當您終止執行個體，或 Auto Scaling 群組停止並終止執行個體的服務，生命週期也隨之結束。

Note

在執行個體啟動後，馬上就會向您收取費用，即使它們尚未提供服務。

下圖顯示在 Amazon EC2 Auto Scaling 生命週期中，執行個體狀態之間的轉換。



擴展

以下擴增事件指揮 Auto Scaling 群組啟動 EC2 執行個體，並將它們連接到群組：

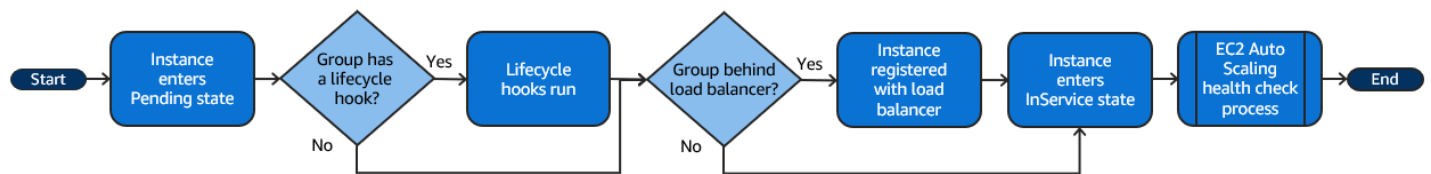
- 您可以手動方式增加群組的大小。如需詳細資訊，請參閱[更改現有 Auto Scaling 群組的所需容量](#)。
- 您可建立一個擴展政策，依照指定的增加需求來自動擴大群組大小。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 動態擴展](#)。
- 您可藉由排程來設定擴展，在特定時間增加群組的大小。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 排程擴展](#)。

當水平擴展事件發生時，Auto Scaling 群組使用其指派的啟動範本來啟動所需數量的 EC2 執行個體。這些執行個體在開始時為 Pending 狀態。如果新增 lifecycle hook 到 Auto Scaling 群組，您可以在這裡執行自訂動作。如需詳細資訊，請參閱[lifecycle hook](#)。

當每個執行個體設定完全且通過 Amazon EC2 運作狀態檢查，就會將它連接到 Auto Scaling 群組並進入 InService 狀態。執行個體會計入 Auto Scaling 群組的所需容量。

如果 Auto Scaling 群組已設定為從 Elastic Load Balancing 負載平衡器接收流量，則 Amazon EC2 Auto Scaling 會自動向負載平衡器註冊執行個體，再將執行個體標記為 InService。

以下摘要說明向負載平衡器註冊執行個體以進行向外擴展事件的步驟。



服務中的執行個體

執行個體會保留組於 InService 狀態，除非以下狀況發生：

- 當發生縮減事件時，則 Amazon EC2 Auto Scaling 會選擇終止此執行個體，以縮減 Auto Scaling 群組的大小。如需詳細資訊，請參閱[控制縮減期間應終止的 Auto Scaling 執行個體](#)。
- 您將執行個體設為 Standby 狀態。如需詳細資訊，請參閱[進入和結束待命](#)。
- 您從 Auto Scaling 群組分開執行個體。如需詳細資訊，請參閱[從 Auto Scaling 群組分離或連接執行個體](#)。
- 執行個體未成功通過運作狀態檢查的數量，所以會從 Auto Scaling 群組移除、終止及取代。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

縮減

以下縮減事件指揮 Auto Scaling 群組將 EC2 執行個體從群組中分開，並將它們終止：

- 您可以手動縮減群組的大小。如需詳細資訊，請參閱[更改現有 Auto Scaling 群組的所需容量](#)。
- 您可建立擴展政策，依照指定的需求縮減自動減少群組大小。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 動態擴展](#)。
- 您可以按照時間表設定擴展，在特定的時間減少群組的大小。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 排程擴展](#)。

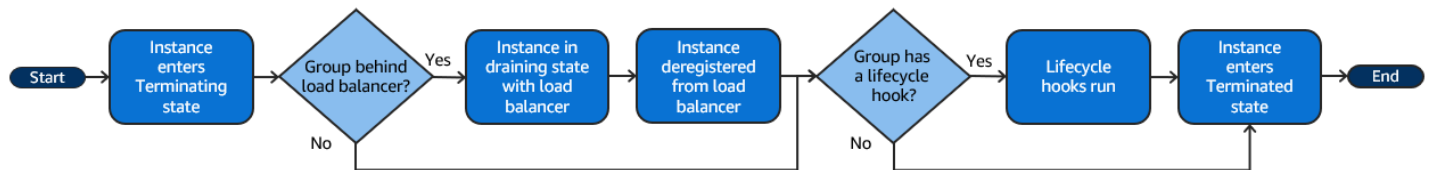
每建立一個擴增事件，就必須建立對應的縮減事件。這有助於確保指派給您的應用程式的資源盡可能符合那些資源的需求。

縮減事件發生時，Auto Scaling 群組會終止一個或多個執行個體。Auto Scaling 群組使用終止政策來決定終止哪一個執行個體。正在從 Auto Scaling 群組終止的執行個體會進入 Terminating 狀態，無法回到服務狀態。

如果 Auto Scaling 群組設定為從 Elastic Load Balancing 負載平衡器接收流量，Amazon EC2 Auto Scaling 會自動從負載平衡器取消註冊正在終止的執行個體。取消註冊執行個體可確保所有新要求都會重新導向至負載平衡器的目標群組中的其他執行個體，同時允許與該執行個體的現有連線繼續，直到取消註冊延遲到期為止。

如果新增 lifecycle hook 到 Auto Scaling 群組，您可以在正在終止的執行個體上執行自訂動作。如需詳細資訊，請參閱[lifecycle hook](#)。最後，執行個體已完全終止，並進入 Terminated 狀態。

以下摘要說明使用負載平衡器取消註冊執行個體以縮減事件的步驟。



分開執行個體

您可以將執行個體從 Auto Scaling 群組分開。在執行個體分開後，您可以分別管理 Auto Scaling 群組和執行個體，或將它連接到不同的 Auto Scaling 群組。

如需詳細資訊，請參閱[從 Auto Scaling 群組分離或連接執行個體](#)。

連接執行個體

您可以連接符合特定條件的執行中 EC2 執行個體至您的 Auto Scaling 群組。在連接執行個體之後，它將以身為 Auto Scaling 群組的一部分受管理。

如需詳細資訊，請參閱[從 Auto Scaling 群組分離或連接執行個體](#)。

lifecycle hook

您可以新增 lifecycle hook 到 Auto Scaling 群組，讓您可以在執行個體啟動或終止時執行自訂動作。

當 Amazon EC2 Auto Scaling 回應擴增事件，它會啟動一或多個執行個體。這些執行個體在開始時為 Pending 狀態。如果您新增 `autoscaling:EC2_INSTANCE_LAUNCHING` lifecycle hook 至

Auto Scaling 群組，此執行個體會從 Pending 狀態變成 Pending:Wait 狀態。在您完成生命週期動作後，此執行個體會進入 Pending:Proceed 狀態。當執行個體完全設定時，它們會連接到 Auto Scaling 群組並進入 InService 狀態。

當 Amazon EC2 Auto Scaling 回應縮減事件，它會終止一或多個執行個體。這些執行個體已從 Auto Scaling 群組分開，並進入 Terminating 狀態。如果您新增 `autoscaling:EC2_INSTANCE_TERMINATING` lifecycle hook 至 Auto Scaling 群組，此執行個體會從 Terminating 狀態變成 Terminating:Wait 狀態。在您完成生命週期動作後，此執行個體會進入 Terminating:Proceed 狀態。當執行個體已完全終止，則會進入 Terminated 狀態。

如需詳細資訊，請參閱 [Amazon EC2 Auto Scaling lifecycle hook](#)。

進入和結束待命

您可以將任何執行個體從 InService 狀態改為 Standby 狀態。這可讓您從服務中移除執行個體、疑難排解或變更，然後回到服務狀態。

處於 Standby 狀態的執行個體，會繼續由 Auto Scaling 群組管理。但是，它們不會成為您的應用程式中使用的一部分，除非您將其返回服務狀態。

如需詳細資訊，請參閱 [從 Auto Scaling 群組暫時移除執行個體](#)。

Auto Scaling 資源和群組的配額

您的 AWS 帳戶 具有每個 AWS 服務的預設配額，先前稱為限制。除非另有說明，否則每個配額都是區域特定的。您可以請求提高某些配額，而其他配額無法提高。

若要檢視 Amazon EC2 Auto Scaling 的配額，請開啟 [Service Quotas console](#) (Service Quotas 主控台)。在導覽窗格中，請選擇 AWS services (服務)，然後選取 Amazon EC2 Auto Scaling。

若要請求提升配額，請參閱《[Service Quotas 使用者指南](#)》中的請求提升配額。若 Service Quotas 中尚未提供配額，請使用 [Auto Scaling 限額表單](#)。提高的配額會繫結到提出請求的區域。

Amazon EC2 Auto Scaling 資源

您的 AWS 帳戶 具有與您可以建立的 Auto Scaling 群組數量和啟動組態相關的下列配額。

資源	預設配額
每個區域的 Auto Scaling 群組	500

資源	預設配額
每個區域的啟動組態	200

Auto Scaling 群組組態

您的 AWS 帳戶 具有與 Auto Scaling 群組組態相關的下列配額。您無法對其進行變更。

資源	配額
每個 Auto Scaling 群組的擴展政策數量	50
每個 Auto Scaling 群組的排程動作數量	125
每個步驟擴展政策的步驟調整數量	20
每個 Auto Scaling 群組的 lifecycle hook 數量	50
每個 Auto Scaling 群組的 SNS 主題數量	10
每個 Auto Scaling 群組的 Classic Load Balancer 數量	50
每個 Auto Scaling 群組的 Elastic Load Balancing 目標群組	50
每個 Auto Scaling 群組的 VPC Lattice 目標群組	5

Auto Scaling 群組 API 操作

Amazon EC2 Auto Scaling 會提供 API 操作，以批次變更 Auto Scaling 群組。以下是單次 API 操作中允許的最大項目數量 (最大陣列成員數量)。您無法對其進行變更。

作業	最大陣列成員數量
AttachInstances	20 個執行個體 ID
AttachLoadBalancers	10 個負載平衡器
AttachLoadBalancerTargetGroups	10 個目標群組

作業	最大陣列成員數量
BatchDeleteScheduledAction	50 個排定的動作
BatchPutScheduledUpdateGroupAction	50 個排定的動作
DetachInstances	20 個執行個體 ID
DetachLoadBalancers	10 個負載平衡器
DetachLoadBalancerTargetGroups	10 個目標群組
EnterStandby	20 個執行個體 ID
ExitStandby	20 個執行個體 ID
SetInstanceProtection	50 個執行個體 ID

Amazon EC2 Auto Scaling API 的請求限流

Amazon EC2 Auto Scaling API 請求會使用字符儲存貯體方案調節，以維持服務頻寬。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 API [請求率](#)。

EC2 終止率

Auto Scaling 群組縮減時，Amazon EC2 Auto Scaling 會動態確定其一次可以執行的 EC2 執行個體終止作業數量。這表示您可能會發現 Auto Scaling 群組同時終止的執行個體數量有所差異。這些差異是因外部考量所造成，例如 Amazon EC2 Auto Scaling 是否必須向負載平衡器取消註冊執行個體。

其他服務

其他服務的配額，例如 Amazon EC2 和 Amazon VPC，可能會影響 Auto Scaling 群組。您可以使用 Service Quotas 來更新 EC2 執行個體和中其他資源的配額 AWS 帳戶。在 Service Quotas 主控台中，您可以檢視所有可用的服務配額，並請求增加服務配額。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的[請求增加配額](#)。

如需啟動範本特定的配額，請參閱《Amazon EC2 使用者指南》中的[啟動範本限制](#)。

設定 以使用 Amazon EC2 Auto Scaling

開始使用 Amazon EC2 Auto Scaling 之前，請先完成以下任務。

任務

- [準備使用 AWS CLI](#)

準備使用 AWS CLI

您可以使用 AWS 命令列工具在系統的命令列發出命令，以執行 Amazon EC2 Auto Scaling 和其他 AWS 任務。

若要使用 AWS Command Line Interface (AWS CLI)，請下載、安裝和設定 版本 1 或 2 AWS CLI。第 1 版和第 2 版中提供相同的 Amazon EC2 Auto Scaling 功能。若要安裝 AWS CLI 第 1 版，請參閱《AWS CLI 第 1 版使用者指南》中的[安裝、更新和解除安裝 AWS CLI](#)。若要安裝第 2 AWS CLI 版，請參閱《第 2 版使用者指南》中的[安裝或更新最新版本的 AWS CLI](#)。AWS CLI

AWS CloudShell 可讓您略過 AWS CLI 在開發環境中安裝，並在 中使用它 AWS 管理主控台。除了避免安裝之外，您也不需要設定憑證，且不需要指定區域。您的 AWS 管理主控台 工作階段會將此內容提供給 AWS CLI。您可以在支援的 AWS CloudShell 中使用 AWS 區域。如需詳細資訊，請參閱[使用從命令列建立 Auto Scaling 群組 AWS CloudShell](#)。

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [autoscaling](#)。

Amazon EC2 Auto Scaling 入門

若要開始使用 Amazon EC2 Auto Scaling，您可以遵循向您介紹服務的教學課程。

主題

- [教學課程：建立您的第一個 Auto Scaling 群組](#)
- [教學課程：設定擴展和負載平衡應用程式](#)

如需著重於管理 Auto Scaling 群組中執行個體生命週期的特定工具的其他教學課程，請參閱下列主題：

- [教學課程：設定叫用 Lambda 函數的 lifecycle hook](#). 本教學課程說明如何使用 Amazon EventBridge 建立規則，以根據 Auto Scaling 群組中執行個體發生的事件調用 Lambda 函數。
- [教學課程：使用資料指令碼和執行個體中繼資料來擷取生命週期狀態](#). 本教學課程說明如何使用執行個體中繼資料服務 (IMDS) 從執行個體本身調用動作。

在建立能與應用程式搭配使用的 Auto Scaling 群組前，請徹底檢閱您於 AWS 雲端中執行的應用程式。考慮下列各項：

- Auto Scaling 群組應跨越多少個可用區域。
- 可使用哪些現有資源，例如安全群組或 Amazon Machine Image (AMI)。
- 無論您想要擴展以增加或減少容量，或是只希望確保永遠執行特定數量的伺服器。請留意，Amazon EC2 Auto Scaling 可同時達成這兩項功能。
- 哪些指標與您的應用程式效能最相關。
- 啟動和佈建伺服器所需的時間。

若您對應用程式更了解，則在您操作 Auto Scaling 架構時，則會更有效率。

教學課程：建立您的第一個 Auto Scaling 群組

本教學課程透過提供 Amazon EC2 Auto Scaling 的實作簡介 AWS 管理主控台。您將建立啟動範本來定義 EC2 執行個體，以及具有單一執行個體的 Auto Scaling 群組。啟動 Auto Scaling 群組後，您將終止執行個體，並確認執行個體已從服務中移除並取代。為了維持固定的執行個體數目，Amazon EC2 Auto Scaling 會自動偵測並回應 Amazon EC2 的運作狀態和連線能力檢查。

註冊時 AWS，您可以使用 [AWS 免費方案](#) 免費開始使用 Amazon EC2 Auto Scaling。您可以使用免費方案來啟動並免費使用 t2.micro 執行個體 12 個月 (在無法使用 t2.micro 的區域中，您可以根據免費方案使用 t3.micro 執行個體)。如果您啟動的執行個體不包含在免費方案內，您將需要為執行個體支付標準 Amazon EC2 使用費。如需詳細資訊，請參閱 [Amazon EC2 定價](#)。

任務

- [準備演練](#)
- [步驟 1：建立啟動範本](#)
- [步驟 2：建立單一執行個體 Auto Scaling 群組](#)
- [步驟 3：驗證 Auto Scaling 群組](#)
- [步驟 4：終止 Auto Scaling 群組中的執行個體](#)
- [步驟 5：後續步驟](#)
- [步驟 6：清除](#)

準備演練

本演練假設您已熟悉如何啟動 EC2 執行個體，且您也已建立金鑰對和安全群組。

若要開始使用 Amazon EC2 Auto Scaling，您可以使用的預設 VPC AWS 帳戶。預設 VPC 在所有可用區域中均包含預設公有子網路，以及連接至您 VPC 的網際網路閘道。您可以在 Amazon Virtual Private Cloud (Amazon VPC) 主控台中 [您的 VPC 頁面](#) 上檢視您的 VPC。

步驟 1：建立啟動範本

在此步驟中，您會建立啟動範本，指定 Amazon EC2 Auto Scaling 為您建立的 EC2 執行個體類型。Amazon EC2 其中的資訊包含：Amazon Machine Image (AMI) 使用的 ID、執行個體類型、金鑰對以及安全群組。

建立啟動範本

1. 開啟 Amazon EC2 主控台並前往 [啟動範本頁面](#)。
2. 在頂部導覽列中選取 AWS 區域。您建立的啟動範本與 Auto Scaling 群組會繫結至您指定的區域。
3. 選擇 Create launch template (建立啟動範本)。
4. 針對 Launch template name (啟動範本名稱)，請輸入 **my-template-for-auto-scaling**。
5. 在 Auto Scaling guidance (Auto Scaling 指引) 下，選取核取方塊。

6. 對於 Application and OS Images (Amazon Machine Image) (應用程式和作業系統映像 (Amazon Machine Image))，請從 Quick Start (快速入門) 清單中選擇 Amazon Linux 2 (HVM) 的版本。AMI 可作為執行個體的基本組態範本。
7. 在 Instance type (執行個體類型) 部分，選擇可與您指定的 AMI 相容之硬體組態。
8. (選用) 針對 Key pair name (金鑰對名稱)，選擇現有的金鑰對。使用金鑰對來以 SSH 連線至 Amazon EC2 執行個體。連接到執行個體未包含在本教學課程中。因此，除非您想要使用 SSH 連線到執行個體，否則不必指定金鑰對。
9. 針對 Network settings (網路設定)，展開 Advanced network configuration (進階網路組態)，並執行下列動作：
 - a. 選擇 Add network interface (新增網路介面) 以設定主要網路介面。
 - b. 針對自動指派公有 IP，指定執行個體是否收到公有 IPv4 地址。根據預設，如果 EC2 Amazon EC2 會指派公有 IPv4 地址。IPv4 如果您不需要連線到執行個體，請選擇停用。
 - c. 針對安全群組 ID，選擇您計劃用作 Auto Scaling 群組之 VPC 的相同 VPC 中的安全群組。若您未指定安全群組，則執行個體會自動與 VPC 的預設安全群組相關聯。
 - d. 針對終止時刪除，選擇是以在刪除執行個體時刪除網路介面。
10. 選擇 Create launch template (建立啟動範本)。
11. 在確認頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

步驟 2：建立單一執行個體 Auto Scaling 群組

使用下列程序在建立啟動範本後，從您離開的地方繼續。

建立 Auto Scaling 群組

1. 在 Choose launch template or configuration (選擇啟動範本或組態) 頁面上，在 Auto Scaling group name (Auto Scaling 群組名稱) 中輸入 **my-first-asg**。
2. 選擇下一步。

選擇執行個體啟動選項頁面隨即出現，可讓您選擇您希望 Auto Scaling 群組使用的 VPC 網路設定，並提供啟動隨需執行個體和 Spot 執行個體的選項。

3. 在網路區段中，將 VPC 設定為您選擇的預設 VPC AWS 區域，或選取您自己的 VPC。預設 VPC 會自動設定為為您的執行個體提供網際網路連線。此 VPC 包含在區域中每個可用區域的公有子網路。

4. 在 Availability Zones and subnets (可用區域和子網路) 中，請從要包含的每個可用區域內選擇子網路。在多個可用區域中，請使用子網路，以獲得高可用性。如需詳細資訊，請參閱[選擇 VPC 子網路時的注意事項](#)。
5. 在 Instance type requirements (執行個體類型需求) 區段中，請使用預設設定來簡化此步驟。(請勿覆寫啟動範本。) 在本教程中，您將僅用啟動範本中指定的執行個體類型僅啟動一個隨需執行個體。
6. 保留本教學課程的其他預設值，然後選擇 Skip to review (跳過以進行檢閱)。

 Note

群組的初始大小取決於所需的容量。預設值為 1 個執行個體。

7. 在 Review (檢閱) 頁面中，檢閱群組的資訊，然後選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

步驟 3：驗證 Auto Scaling 群組

建立 Auto Scaling 群組後，您就可以驗證該群組是否已經啟動 EC2 執行個體。

 Tip

在下列程序中，您可以查看 Auto Scaling 群組的 Activity history (活動歷史記錄) 和 Instances (執行個體) 部分。在這兩個部分中，已顯示了指定的資料欄。要顯示隱藏的資料欄列或變更顯示的資料列，請選擇每個部分右上角的齒輪圖示以開啟偏好設定模式，視需要更新設定，然後選擇 Confirm (確認)。

驗證 Auto Scaling 群組已啟動 EC2 執行個體

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取您剛建立的 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。第一個可用的索引標籤是 Details (詳細資訊) 索引標籤，其上會顯示 Auto Scaling 群組的資訊。

3. 選擇第二個索引標籤，Activity (活動)。在 Activity history (活動歷史記錄) 中，您可以檢視與 Auto Scaling 群組相關聯的活動進度。Status (狀態) 欄顯示執行個體的目前狀態。在執行個體啟動的同

時，狀態欄顯示 Not yet in service。在執行個體啟動後，狀態會變更成 Successful。您也可以使用重新整理按鈕，查看執行個體的目前狀態。

4. 在 Instance management (執行個體管理) 索引標籤的 Instances (執行個體) 下方，您可以檢視執行個體的状态。
5. 確認您已成功啟動執行個體。啟動執行個體無須費時。
 - Lifecycle (生命週期) 欄顯示執行個體的状态。初始時，您的執行個體為 Pending 状态。在執行個體準備好接受流量之後，它的状态會變成 InService。
 - 運作状态欄會顯示執行個體上 Amazon EC2 Auto Scaling 運作状态檢查的結果。

步驟 4：終止 Auto Scaling 群組中的執行個體

您可以使用這些步驟來進一步了解 Amazon EC2 Auto Scaling 如何運作，尤其是在必要時如何啟動新的執行個體。您在本教學課程中建立的 Auto Scaling 群組規模下限為一個執行個體。因此，若您終止執行中的執行個體，則 Amazon EC2 Auto Scaling 必須啟動新的執行個體以將其取代。

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組旁的核取方塊。
3. 在 Instance management (執行個體管理) 索引標籤的 Instances (執行個體) 下方，選取執行個體的 ID。

這會將您引導至 Amazon EC2 主控台的 Instances (執行個體) 頁面，您可以在該頁面終止執行個體。

4. 依序選擇 Actions (動作)、Instance State (執行個體状态) 和 Terminate (終止)。出現確認提示時，請選擇 Yes, Terminate (是，終止)。
5. 在導覽窗格的 Auto Scaling 下，選擇 Auto Scaling Groups (Auto Scaling 群組)。選取您的 Auto Scaling 群組，然後選擇 Activity (活動) 索引標籤。

當您從執行個體頁面終止執行個體時，在終止執行個體之後，需要一到兩分鐘才能啟動新的執行個體。在活動歷史記錄中，當擴展活動開始時，您會看到第一個執行個體終止項目與新的執行個體啟動項目。使用重新整理按鈕，直到您看到新的項目為止。

6. 在 Instance management (執行個體管理) 索引標籤上，Instances (執行個體) 區段僅會顯示新的執行個體。
7. 在導覽窗格的 Instances (執行個體) 下方，選擇 Instances (執行個體)。此頁面會同時顯示終止的執行個體與執行中的新執行個體。

步驟 5：後續步驟

如果您想要刪除剛建立的基本基礎設施，請前往下一個步驟。否則，您可以使用此基礎設施做為基礎，嘗試一或多個下列選項：

- 使用 Session Manager 或 SSH 連線到您的 Linux 執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[使用 Session Manager 連線至 EC2 執行個體](#)和[使用 SSH 連線至 Linux 執行個體](#)。Amazon EC2
- 設定 Amazon SNS 通知，以在 Auto Scaling 群組啟動或終止執行個體時通知您。如需詳細資訊，請參閱[Amazon SNS 通知選項](#)。
- 手動擴展 Auto Scaling 群組以測試 SNS 通知。如需詳細資訊，請參閱[更改您的 Auto Scaling 群組的所需容量](#)。

您也可以閱讀 [目標追蹤擴展政策](#)，開始自行熟悉自動擴展概念。如果應用程式的負載發生變化，Auto Scaling 群組可以在最小和最大容量限制之間調整所需的群組容量，從而自動擴充 (新增執行個體) 和縮減 (執行較少執行個體)。如需有關設定這些限制的詳細資訊，請參閱 [設定 Auto Scaling 群組的擴展限制](#)。

步驟 6：清除

您可以刪除擴展基礎設施，或僅刪除 Auto Scaling 群組，並保留啟動範本以供稍後使用。

若您啟動的執行個體不在 [AWS 免費方案](#)內，您需要終止該執行個體以免產生額外費用。當您終止執行個體時，與其相關聯的資料也會一併刪除。

若要刪除 Auto Scaling 群組

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組 (my-first-asg) 旁的核取方塊。
3. 選擇 刪除。
4. 出現確認提示時，請輸入 **delete** 來確認刪除特定的 Auto Scaling 群組，然後選擇 Delete (刪除)。

Name (名稱) 欄位中的載入圖示會顯示正在刪除 Auto Scaling 群組。刪除過程中，Desired (所需)、Min (最小) 和 Max (最大) 欄位會顯示 Auto Scaling 群組的 0 個執行個體。終止執行個體並刪除群組需要幾分鐘的時間。重新整理清單，以查看目前狀態。

若想保留您的啟動範本，請跳過以下程序。

刪除您的啟動範本

1. 開啟 Amazon EC2 主控台中的[啟動範本頁面](#)。
2. 選取您的啟動範本 (my-template-for-auto-scaling)。
3. 選擇 Actions (動作)、Delete template (刪除範本)。
4. 出現確認提示時，請輸入 **Delete** 確認刪除指定的啟動範本，然後選擇 Delete (刪除)。

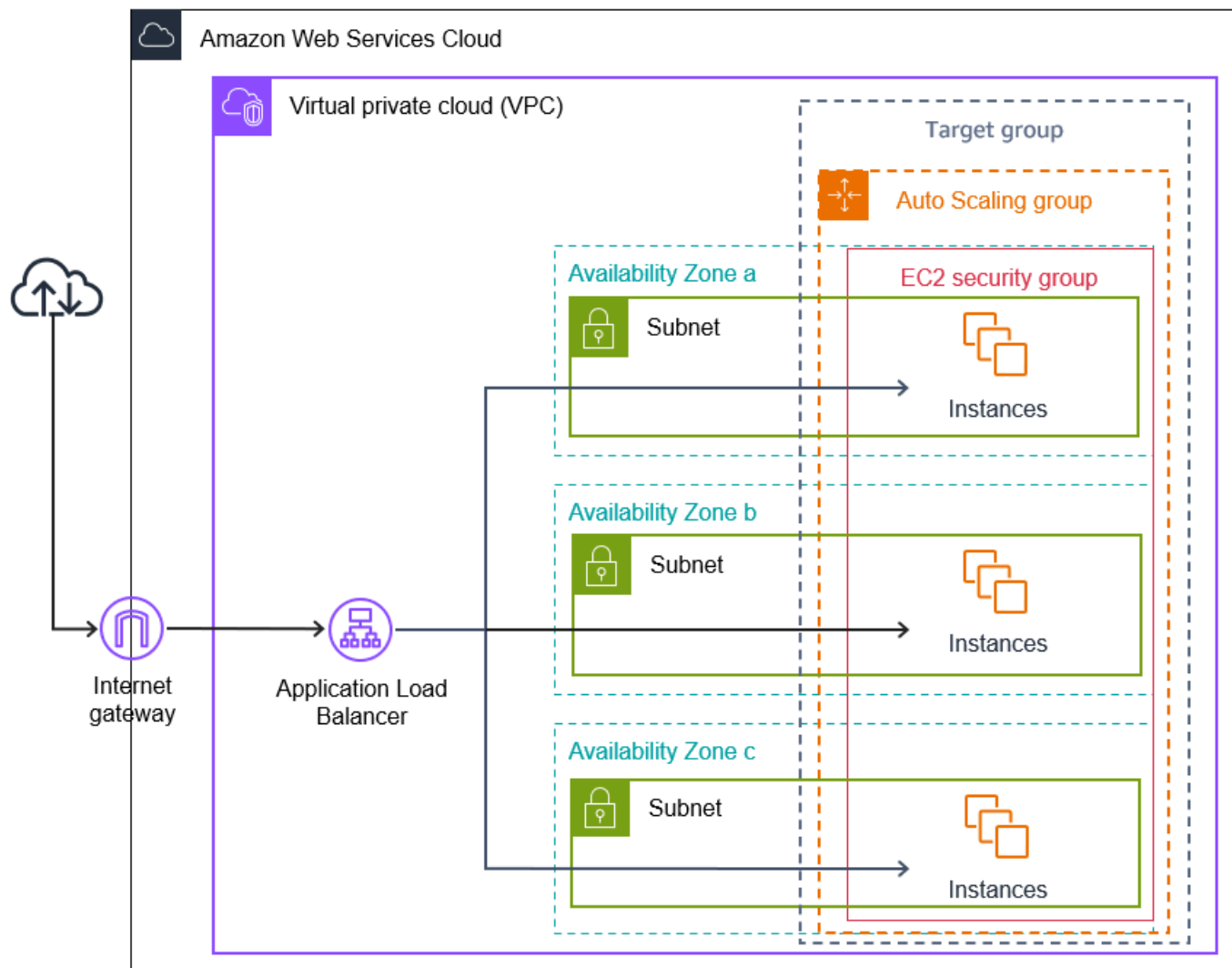
教學課程：設定擴展和負載平衡應用程式

Important

在探索本教學課程之前，建議您先檢閱下列簡介教學課程：[建立您的第一個 Auto Scaling 群組](#)。

使用 Elastic Load Balancing 負載平衡器註冊 Auto Scaling 群組有助於您設定負載平衡應用程式。Elastic Load Balancing 可搭配 Amazon EC2 Auto Scaling 使用，將傳入流量分配到狀態良好的 Amazon EC2 執行個體。進而提升應用程式的可擴展性與可用性。您可以在多個可用區域內啟用 Elastic Load Balancing，以提高應用程式的容錯能力。

在本教學課程中，我們會詳細說明在建立 Auto Scaling 群組時設定負載平衡應用程式的基本步驟。完成後，您的架構看起來應該會如下圖所示：



Elastic Load Balancing 支援多種不同類型的負載平衡器。建議您在本教學課程中使用 Application Load Balancer。

如需關於將負載平衡器引入架構的詳細資訊，請參閱 [使用 Elastic Load Balancing 在 Auto Scaling 群組中分配傳入應用程式流量](#)。

任務

- [先決條件](#)
- [步驟 1：設定啟動範本或啟動組態](#)
- [步驟 2：建立 Auto Scaling 群組](#)
- [步驟 3：驗證負載平衡器是否已連接](#)
- [步驟 4：後續步驟](#)

- [步驟 5：清除](#)
- [相關資源](#)

先決條件

- 負載平衡器和目標群組。務必為計劃用於 Auto Scaling 群組的負載平衡器選擇相同的可用區域。如需詳細資訊，請參閱《Elastic Load Balancing 使用者指南》中的 [Elastic Load Balancing 入門](#)。
- 啟動範本或啟動組態的安全群組。安全群組必須允許從接聽程式連接埠 (通常是 HTTP 流量的連接埠 80) 上的負載平衡器存取，以及您要 Elastic Load Balancing 用於運作狀態檢查的連接埠上的負載平衡器存取。如需詳細資訊，請參閱適用的文件：
 - 《Application Load Balancer 使用者指南》中的 [目標安全群組](#)。
 - 《Network Load Balancer 使用者指南》中的 [目標安全群組](#)。

若您的執行個體將會有公有 IP 地址，您可以允許 SSH 流量連線到執行個體。

- (選用) 授予應用程式存取權的 IAM 角色 AWS。
- (選用) 將 Amazon Machine Image (AMI) 定義為 Amazon EC2 執行個體的來源範本。若要立即建立，請啟動執行個體。接著，您必須指定 IAM 角色 (如有建立的話) 和需要作為使用者資料的任何組態指令碼。連線到執行個體並加以自訂。例如，您可以安裝軟體和應用程式，複製資料，以及連接額外的 EBS 磁碟區。在您的執行個體上測試應用程式，確保設定正確無誤。您可以將這個更新過的組態儲存為自訂 AMI。如果之後不需要執行個體，您可以終止執行個體。從這個新自訂 AMI 啟動的執行個體包括您建立 AMI 時所設的自訂。
- Virtual Private Cloud (VPC)。本教學課程使用的是預設 VPC，但您可以使用自己的 VPC。在使用自己的 VPC 的情況下，請確定您的 VPC 有一個子網路映射到您正在使用之區域的每個可用區域。必須至少有兩個可用的公有子網路，您才能建立負載平衡器。也必須具有兩個私有子網路或兩個公有子網路，您才能建立 Auto Scaling 群組並使用負載平衡器註冊。

步驟 1：設定啟動範本或啟動組態

使用本教學課程的啟動範本或啟動組態。

主題

- [選取或建立啟動範本](#)
- [選擇或建立啟動組態](#)

選取或建立啟動範本

如果您已經有偏好使用的啟動範本，則請依照下列操作程序進行選擇。

選擇現有的啟動範本

1. 開啟 Amazon EC2 主控台中的[啟動範本頁面](#)。
2. 在畫面頂端的導覽列上，選擇建立負載平衡器的區域。
3. 選擇啟動範本。
4. 選擇 Actions (動作)、Create an Auto Scaling group (建立 Auto Scaling 群組)。

或者，可依照下列步驟建立新的啟動範本。

建立啟動範本

1. 開啟 Amazon EC2 主控台中的[啟動範本頁面](#)。
2. 在畫面頂端的導覽列上，選擇建立負載平衡器的區域。
3. 選擇 Create launch template (建立啟動範本)。
4. 為初始版本的啟動範本輸入名稱並提供描述。
5. 對於 Application and OS Images (Amazon Machine Image) (應用程式和作業系統映像 (Amazon Machine Image))，為您的執行個體選擇 AMI 的 ID。您可以搜尋所有可用的 AMI，或從 Recent (最近) 或 Quick Start (快速開始) 清單中選取 AMI。若未看到您需要的 AMI，請選擇 Browser more AMIs (瀏覽更多 AMI) 即可瀏覽完整的 AMI 目錄。
6. 針對 Instance type (執行個體類型)，選擇可與您指定的 AMI 相容之執行個體硬體組態。
7. (選用) 針對 Key pair (login) (金鑰對 (登入))，選擇要在連線執行個體時使用的金鑰對。
8. 針對 Network settings (網路設定)，展開 Advanced network configuration (進階網路組態)，並執行下列動作：
 - a. 選擇 Add network interface (新增網路介面) 以設定主要網路介面。
 - b. 對於自動指派公有 IP，請指定您的執行個體是否接收公有 IPv4 地址。根據預設，如果 EC2 Amazon EC2 會指派公有 IPv4 地址。IPv4 如果您不需要連線到執行個體，您可以選擇停用，以防止群組中的執行個體直接從網際網路接收流量。在此情況下，它們只會從負載平衡器接收流量。
 - c. 針對 Security group ID (安全群組 ID)，從與負載平衡器相同的 VPC 為執行個體指定安全群組。

- d. 針對 Delete on termination (在終止時刪除)，選擇 Yes (是)。這會在 Auto Scaling 群組縮減時刪除網路介面，並終止網路介面連接的執行個體。
9. (選用) 若要安全地分配憑證到您的執行個體，針對 Advanced details (進階詳細資訊)、IAM instance profile (IAM 執行個體描述檔)，輸入 IAM 角色的 Amazon Resource Name (ARN)。
10. (選用) 若要為執行個體指定使用者資料或組態指令碼，請於 Advanced details (進階詳細資訊) 與 User data (使用者資料) 貼上使用者資料或組態指令碼。
11. 選擇 Create launch template (建立啟動範本)。
12. 在確認頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

選擇或建立啟動組態

Note

我們強烈建議在新應用程式中使用啟動組態，因為它是沒有計劃投資的舊版功能。此外，在 2023 年 6 月 1 日當天或之後建立的新帳戶將無法選擇透過主控台建立新的啟動組態。如需詳細資訊，請參閱[Auto Scaling 啟動組態](#)。

選擇現有的啟動組態

1. 開啟 Amazon EC2 主控台中的[啟動組態頁面](#)。
2. 在頂端導覽列上，選擇建立負載平衡器的區域。
3. 選擇啟動組態。
4. 選擇 Actions (動作)、Create an Auto Scaling group (建立 Auto Scaling 群組)。

或者，您可以使用下列操作程序來建立新的啟動組態。

建立啟動組態

1. 開啟 Amazon EC2 主控台中的[啟動組態頁面](#)。出現確認提示時，請選擇檢視啟動組態，以確認您要檢視啟動組態頁面。
2. 在頂端導覽列上，選擇建立負載平衡器的區域。
3. 選擇 Create launch configuration (建立啟動組態)，然後輸入啟動組態的名稱。
4. 針對 Amazon machine image (AMI)，請將執行個體 AMI 的 ID 輸入為搜尋條件。
5. 針對 Instance type (執行個體類型)，為執行個體選取硬體組態。

6. 在 Additional configuration (其他組態) 下，請注意下列欄位：
 - a. (選用) 若要安全地分配憑證給 EC2 執行個體，請就 IAM 執行個體描述檔選擇您的 IAM 角色。如需詳細資訊，請參閱[在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)。
 - b. (選用) 若要為執行個體指定使用者資料或組態指令碼，請於 Advanced details (進階詳細資訊) 與 User data (使用者資料) 貼上使用者資料或組態指令碼。
 - c. (選用) 針對 Advanced Details (進階詳細資訊)、IP address Type (IP 地址類型)，請保留預設值。建立 Auto Scaling 群組時，可以使用已啟用公有 IP 地址屬性的子網路 (例如預設 VPC 中的預設子網路)，將公有 IP 地址指派給 Auto Scaling 群組中的執行個體。或者，如果您不需要連線到執行個體，可以選擇 Do not assign a public IP address to any instances (不要將公有 IP 地址指派給任何執行個體)，以防止群組中的執行個體直接從網際網路接收流量。在此情況下，它們只會從負載平衡器接收流量。
7. 針對 Security groups (安全群組)，請從與負載平衡器相同的 VPC 中選擇現有安全群組。如果選擇 Create a new security group (建立新的安全群組) 選項，則會為 Amazon EC2 Linux 執行個體設定預設 SSH 規則。為 Amazon EC2 Windows 執行個體設定預設 RDP 規則。
8. 針對 Key pair (login) (金鑰對 (登入))，請選擇 Key pair options (金鑰對選項) 下的某個選項。

若您已經設定 Amazon EC2 執行個體金鑰對，則可以在這裡選擇它。

如果您還沒有 Amazon EC2 執行個體金鑰對，則請選擇 Create a new key pair (建立新的金鑰對)，然後給予可輕鬆辨識的名稱。選擇 Download Key Pair (下載金鑰對)，將金鑰對下載至電腦。

Important

如果您需要連線到執行個體，請勿選取 Proceed without a key pair (沒有金鑰對仍繼續作業)。

9. 選取確認核取方塊，然後選擇 Create launch configuration (建立啟動組態)。
10. 選取新啟動組態名稱旁的核取方塊，然後依序選擇 Actions (動作)、Create Auto Scaling group (建立 Auto Scaling 群組)。

步驟 2：建立 Auto Scaling 群組

您可以使用下列操作程序，接續您建立或選取啟動範本或啟動組態後未繼續作業的階段。

建立 Auto Scaling 群組

1. 在選擇啟動範本或組態頁面上，針對 Auto Scaling 群組名稱，請輸入 Auto Scaling 群組的名稱。
2. [僅限啟動範本] 針對 Launch template (啟動範本)，選擇 Auto Scaling 群組擴增時使用啟動範本的預設、最新或特定版本。
3. 選擇下一步。

Choose instance launch options (選擇執行個體啟動選項) 頁面隨即顯示，可讓您選擇想要 Auto Scaling 群組使用的 VPC 網路設定，並為您提供啟動隨需執行個體和 Spot 執行個體的選項 (如果您選擇啟動範本)。

4. 在 Network (網路) 區段中，對於 VPC，請選擇您用於負載平衡器的 VPC。如果您選擇預設 VPC，它會自動設定為為您的執行個體提供網際網路連線。此 VPC 包含在區域中每個可用區域的公有子網路。
5. 對於 Availability Zones and subnets (可用區域和子網路)，根據負載平衡器所在的可用區域，從要包含的每個可用區域中選擇一或多個子網路。如需詳細資訊，請參閱[選擇 VPC 子網路時的注意事項](#)。
6. [僅限啟動範本] 在 Instance type requirements (執行個體類型需求) 區段中，使用預設的設定簡化此步驟。(請勿覆寫啟動範本。) 在本教程中，您將使用啟動範本中指定的執行個體類型，並且僅啟動隨需執行個體。
7. 選擇 Next (下一頁) 前往 Configure advanced options (設定進階選項) 頁面。
8. 要將群組連接至現有負載平衡器，請在 Load balancing (負載平衡) 區段中，選擇 Attach to an existing load balancer (連接至現有負載平衡器)。您可以選擇 Choose from your load balancer target groups (從負載平衡器目標群組中選擇) 或者 Choose from Classic Load Balancers (從 Classic Load Balancer 選擇)。然後，您可以為您建立的 Application Load Balancer 或 Network Load Balancer 選擇目標群組名稱，或選擇 Classic Load Balancer 的名稱。
9. (選用) 在運作狀態檢查、其他運作狀態檢查類型上，選取開啟 Elastic Load Balancing 運作狀態檢查。
10. (選用) 針對運作狀態檢查寬限期，輸入時長，以秒為單位。此時間量是 Amazon EC2 Auto Scaling 在進入 InService 狀態後檢查執行個體的運作狀態所需的等待時間。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。
11. 當您完成對 Auto Scaling 群組的設定後，請選擇 Skip to review (跳過以進行檢閱)。
12. 詳閱 Review (檢閱) 頁面上的 Auto Scaling 群組詳細資訊。您可以選擇 Edit (編輯) 來進行變更。完成時，選擇 Create Auto Scaling group (建立群組)。

在您將負載平衡器連接到建立的 Auto Scaling 群組後，該負載平衡器即會在新執行個體上線時自動為其註冊。您目前只有一個執行個體，因此沒有太多註冊作業需要進行。不過，您能夠更新群組所需的容量，以新增其他執行個體。如需逐步說明，請參閱 [更改您的 Auto Scaling 群組的所需容量](#)。

步驟 3：驗證負載平衡器是否已連接

驗證負載平衡器是否已連接

1. 從 Amazon EC2 主控台的 [Auto Scaling groups](#) (Auto Scaling 群組) 頁面中，選取 Auto Scaling 群組旁的核取方塊。
2. 在 Details (詳細資訊) 索引標籤上，Load balancing (負載平衡) 會顯示所有已連接的負載平衡器目標群組或 Classic Load Balancer。
3. 在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 中，您可以確認執行個體是否成功啟動。Status (狀態) 欄位會顯示 Auto Scaling 群組是否成功地啟動了執行個體。如果執行個體無法啟動，您可以在 [對 Amazon EC2 Auto Scaling 中的問題進行故障診斷](#) 中找到常見執行個體啟動問題的故障診斷想法。
4. 在 Instance management (執行個體管理) 索引標籤的 Instances (執行個體) 下，您可以確認執行個體是否準備好接受流量。一開始，您的執行個體會處於 Pending 狀態。在執行個體準備好接受流量之後，它的狀態會變成 InService。Health status (運作狀態) 欄位會顯示執行個體的 Amazon EC2 Auto Scaling 運作狀態檢查結果。雖然執行個體可能標記為狀態良好，但負載平衡器只會將流量傳送到通過負載平衡器運作狀態檢查的執行個體。
5. 確認您的執行個體已在負載平衡器上註冊。開啟 Amazon EC2 主控台中的 [目標群組頁面](#)。選取您的目標群組，然後選擇 Targets (目標) 索引標籤。如果執行個體的状态為 initial，這可能是因為它們仍在註冊過程中，或者它們仍在進行運作狀態檢查。當您執行個體的状态為 healthy 時，表示已準備好可供使用。

步驟 4：後續步驟

現在您已經完成此教學課程，您可以進一步了解：

- Amazon EC2 Auto Scaling 會根據您的 Auto Scaling 群組使用的運作狀態檢查狀態，判斷執行個體是否運作狀態良好。如果您啟用負載平衡器運作狀態檢查，且執行個體未通過運作狀態檢查，Auto Scaling 群組會將執行個體視為運作狀態不佳，並予以取代。如需詳細資訊，請參閱 [運作狀態檢查](#)。
- 您可將應用程式擴展到相同區域的另一個可用區域，藉此提高容錯能力以因應服務中斷情況。如需詳細資訊，請參閱 [新增可用區域](#)。

- 您可以將 Auto Scaling 群組設定為使用目標追蹤擴展政策。這會在執行個體需求變更時自動增加或減少執行個體數目。這可讓群組處理應用程式接收的流量變更。如需詳細資訊，請參閱[目標追蹤擴展政策](#)。

步驟 5：清除

在完成為本教學課程建立的資源後，您應該考慮清理這些資源，以避免產生不必要的費用。

若要刪除 Auto Scaling 群組

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組旁的核取方塊。
3. 選擇 刪除。
4. 出現確認提示時，請輸入 **delete** 來確認刪除特定的 Auto Scaling 群組，然後選擇 Delete (刪除)。

Name (名稱) 欄位中的載入圖示會顯示正在刪除 Auto Scaling 群組。刪除過程中，Desired (所需)、Min (最小) 和 Max (最大) 欄位會顯示 Auto Scaling 群組的 0 個執行個體。終止執行個體並刪除群組需要幾分鐘的時間。重新整理清單，以查看目前狀態。

若想保留您的啟動範本，請跳過以下程序。

刪除您的啟動範本

1. 開啟 Amazon EC2 主控台內的 [啟動範本頁面](#)。
2. 選擇啟動範本。
3. 選擇 Actions (動作)、Delete template (刪除範本)。
4. 出現確認提示時，請輸入 **Delete** 確認刪除指定的啟動範本，然後選擇 Delete (刪除)。

若想保留您的啟動組態，請跳過以下程序。

刪除您的啟動組態

1. 開啟 Amazon EC2 主控台內的 [啟動組態頁面](#)。
2. 選取啟動組態。
3. 選擇 Actions (動作)、Delete launch configuration (刪除啟動組態)。

4. 出現確認提示時，請選擇刪除。

若想保留負載平衡器以供日後使用，請略過下列操作程序。

刪除負載平衡器

1. 開啟 Amazon EC2 主控台中的[負載平衡器頁面](#)。
2. 選擇負載平衡器，然後選擇 Actions (動作)、Delete (刪除)。
3. 出現確認提示時，選擇 Yes, Delete (是，刪除)。

若要刪除您的目標群組

1. 開啟 Amazon EC2 主控台中的[目標群組頁面](#)。
2. 選取目標群組，然後依序選擇 Actions (動作)、Delete (刪除)。
3. 出現確認提示時，選擇 Yes, Delete (是，刪除)。

相關資源

透過 CloudFormation，您可以使用範本檔案以單一單位 (堆疊) 的形式建立和刪除資源集合，以可預測且重複的方式建立和佈建 AWS 基礎設施部署。如需詳細資訊，請參閱「[AWS CloudFormation 使用者指南](#)」。

如需示範如何使用堆疊範本來佈建 Auto Scaling 群組和 Application Load Balancer，請參閱《AWS CloudFormation 使用者指南》中的[演練：建立可擴展且負載平衡的應用程式](#)。使用此演練和範例範本作為建立類似範本的起點，以符合您的需求。

Auto Scaling 啟動範本

啟動範本與[啟動組態](#)類似，都會指定執行個體組態資訊。其中包含 Amazon Machine Image (AMI) 的 ID、執行個體類型、金鑰對、安全群組，以及您用來啟動 EC2 執行個體的其他參數。然而，定義啟動範本 (而非啟動組態) 能讓您擁有多個版本的啟動範本。

藉由啟動範本的版本控制，您可以建立一組完整的參數子集，之後可以重複使用子集來建立相同啟動範本的其他版本。例如：您可以建立啟動範本來定義不具備 AMI 或使用者資料指令碼的基本組態。建立啟動範本後，您可以建立新版本，再新增具備最新版應用程式的 AMI 和使用者資料來進行測試。此操作會使啟動範本呈現兩種版本。存放基本組態可協助您維護必要的一般組態參數。您可以隨時從基本組態建立啟動範本的新版本。如果不再需要，也可以將用來測試應用程式的版本刪除。

建議您使用啟動範本，確保可以存取最新功能和改善項目。即便使用啟動組態，也並非所有 Amazon EC2 Auto Scaling 功能都可用。例如：您無法建立同時啟動 Spot 及隨需執行個體的 Auto Scaling 群組或指定多個執行個體類型的群組。您必須使用啟動範本來設定這些功能。如需詳細資訊，請參閱 [具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

若使用啟動範本，您也可以使用 Amazon EC2 的較新功能。這包括 Systems Manager 參數 (AMI ID)、目前世代的 EBS 佈建 IOPS 磁碟區 (io2)、EBS 磁碟區標記、T2 無限制執行個體、容量保留 Capacity Blocks、和 專用主機等。

在建立啟動範本時，所有參數都是選用參數。因此，若啟動範本未指定 AMI，您便無法在建立 Auto Scaling 群組時新增 AMI。若您已指定 AMI 但未指定執行個體類型，則可以在建立 Auto Scaling 群組時新增一或多個執行個體類型。

目錄

- [使用啟動範本的許可](#)
- [啟動範本支援的 API 操作](#)
- [為 Auto Scaling 群組建立啟動範本](#)
- [使用進階設定建立啟動範本](#)
- [遷移 Auto Scaling 群組以啟動範本](#)
- [遷移 AWS CloudFormation 堆疊以啟動範本](#)
- [使用 建立和管理啟動範本的範例 AWS CLI](#)
- [在啟動範本中使用 AWS Systems Manager 參數而非 AMI IDs](#)

使用啟動範本的許可

本節中的程序假設您已擁有建立啟動範本的必要許可。如需有關管理員如何授予您許可的資訊，請參閱《Amazon EC2 使用者指南》中的[使用 IAM 許可控制啟動範本的存取權](#)。

請注意，如果您沒有足夠的許可來使用和建立啟動範本中指定的資源，當您嘗試為 Auto Scaling 群組指定啟動範本時，您會收到錯誤通知，指示您無權使用啟動範本。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 的故障診斷：啟動範本](#)。

如需可讓您使用啟動範本呼叫 CreateAutoScalingGroup、UpdateAutoScalingGroup 和 RunInstances API 操作的 IAM 政策範例，請參閱 [在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量](#)。

啟動範本支援的 API 操作

如需啟動範本支援的 API 操作清單，請參閱《[Amazon EC2 API 參考](#)》中的 [Amazon EC2 動作](#)。

為 Auto Scaling 群組建立啟動範本

在可以使用啟動範本建立 Auto Scaling 群組之前，您必須建立啟動範本，其中含有啟動執行個體所需的組態資訊，包括 Amazon Machine Image (AMI) 的 ID。

要建立新的啟動範本，請依照下列程序：

目錄

- [建立啟動範本 \(主控台\)](#)
- [變更預設網路介面設定 \(主控台\)](#)
- [修改儲存組態 \(主控台\)](#)
- [從現有執行個體建立啟動範本 \(主控台\)](#)
- [相關資源](#)
- [限制](#)

Important

在您建立啟動範本時，啟動範本參數尚未完整驗證。如果您為參數指定了不正確的數值，或是未使用支援的參數組合，則無法使用此啟動範本啟動任何執行個體。請務必指定正確的參

數值，並使用受支援的參數組合。例如，若要使用 ARM 型 AWS Graviton 或 Graviton2 AMI 來啟動執行個體，就必須指定 ARM 兼容的執行個體類型。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[啟動範本限制](#)。

建立啟動範本 (主控台)

下列為設定基本啟動範本的詳細步驟：

- 請指定要從中啟動執行個體的 Amazon Machine Image (AMI)。
- 請選擇與您指定的 AMI 相容的執行個體類型。
- 指定連線至執行個體時要使用的金鑰對，例如：使用 SSH。
- 新增一個或多個安全群組，以便允許對執行個體進行網路存取。
- 請指定是否要連接其他磁碟區至每個執行個體。
- 將自訂標籤 (鍵值對) 新增至執行個體和磁碟區。

建立啟動範本

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在導覽窗格的 Instances (執行個體) 底下，選擇 Launch Templates (啟動範本)。
3. 選擇 Create launch template (建立啟動範本)。為初始版本的啟動範本輸入名稱並提供描述。
4. (選用) 在 Auto Scaling 指引下，請勾選核取方塊以藉 Amazon EC2 提供指引，協助建立要搭配 Amazon EC2 Auto Scaling 使用的範本。
5. 在 Launch template contents (啟動範本內容) 下，根據需要填寫每個必要欄位以及任何選用欄位。
 - a. 應用程式和作業系統映像 (Amazon Machine Image)：(必填) 請為執行個體選擇 AMI 的 ID。您可以搜尋所有可用的 AMI，或從 Recent (最近) 或 Quick Start (快速開始) 清單中選取 AMI。若未看到您需要的 AMI，請選擇 Browser more AMIs (瀏覽更多 AMI) 即可瀏覽完整的 AMI 目錄。

若要選擇自訂 AMI，必須先從自訂的執行個體建立 AMI。如需詳細資訊，請參閱 [《Amazon EC2 使用者指南》中的建立 Amazon EBS 後端 AMI](#)。Amazon EC2
 - b. 在 Instance type (執行個體類型) 欄位中，選擇與您指定之 AMI 相容的單一執行個體類型。

另外，若要使用屬性型執行個體類型選取，請選擇進階、指定執行個體類型屬性，並指定下列選項：

- Number of vCPUs (vCPU 數量)：輸入 vCPU 數量下限和上限。若要表示無限制，請在下限中輸入 0，將上限保留空白。
 - Amount of memory (MiB) (記憶體數量 (MiB))：輸入記憶體數量 (MiB) 下限和上限。若要表示無限制，請在下限中輸入 0，將上限保留空白。
 - 展開 Optional instance type attributes (選用執行個體類型屬性)，然後選擇 Add attribute (新增屬性)，以進一步限制可用於滿足所需容量的執行個體類型。如需每個屬性的詳細資訊，請參閱《Amazon EC2 API 參考》中的 [InstanceRequirementsRequest](#)。
 - 結果執行個體類型：您可以檢視符合指定運算需求的執行個體類型，例如 vCPUs、記憶體和儲存。
 - 若要排除執行個體類型，請選擇 Add attribute (新增屬性)。從 Attribute (屬性) 清單中，選擇 Excluded instance types (已排除的執行個體類型)。從 Attribute value (屬性值) 清單中，選取要排除的執行個體類型。
- c. 金鑰對 (登入)：在 Key pair name (金鑰對名稱) 欄位中，選擇現有的金鑰對，或選擇 Create new key pair (建立新金鑰對) 以建立新的金鑰對。如需詳細資訊，請參閱 [《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和 Linux 執行個體](#)。Amazon EC2
- d. 網路設定：在 Firewall (Security Groups) (防火牆 (安全群組)) 欄位中，選擇一個或多個安全群組或保留空白，並設定一或多個安全群組作為網路介面的一部分。如需詳細資訊，請參閱 [《Amazon EC2 使用者指南》中的適用於 Linux 執行個體的 Amazon EC2 安全群組](#) 一節。
- 若您未在啟動範本中指定任何安全群組，則 Amazon EC2 會為 VPC (Auto Scaling 群組將於其中啟動執行個體) 使用預設安全群組。根據預設，此安全群組不允許來自外部網路的傳入流量。如需更多資訊，請參閱《Amazon VPC 使用者指南》中的 [VPC 的預設安全群組](#)。
- e. 執行以下任意一項：
- 變更預設網路介面設定。例如，您可以啟用或停用公有 IPv4 定址功能，該功能將覆蓋子網路上的自動指派公有 IPv4 地址設定。如需詳細資訊，請參閱 [變更預設網路介面設定 \(主控台\)](#)。
 - 請略過此步驟，以保留預設網路介面設定。
- f. 執行下列任意一項：
- 修改儲存組態。如需詳細資訊，請參閱 [修改儲存組態 \(主控台\)](#)。
 - 請略過此步驟，以保留預設儲存組態。
- g. 在 Resource tags (資源標籤) 欄位中，透過提供鍵值組合來指定標籤。如果在啟動範本中指定執行個體標籤，接著選擇將 Auto Scaling 群組的標籤傳播到其執行個體，則會合併所有標

籤。如果為啟動範本中的標籤和 Auto Scaling 群組中的標籤指定了相同的標籤鍵，則以群組中的標籤值為優先。

6. (選用) 設定進階設定。例如：您可以選擇在應用程式存取其他 AWS 資源時可以使用的 IAM 角色，或指定在執行個體啟動後用來執行一般自動化組態任務的執行個體使用者資料。如需詳細資訊，請參閱[使用進階設定建立啟動範本](#)。
7. 在您準備好建立啟動範本時，請選擇 Create launch template (建立啟動範本)。
8. 若要建立 Auto Scaling 群組，請從確認頁面中選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

變更預設網路介面設定 (主控台)

網路介面可讓您連線到 VPC 和網際網路中的其他資源。如需詳細資訊，請參閱[使用 Amazon VPC 為您的 Auto Scaling 執行個體提供網際網路連線](#)。

本節介紹如何變更預設網路介面設定。例如：您可以定義是否要將公有 IPv4 地址指派給每個執行個體，而不是採用預設的子網路上自動指派公有 IPv4 地址設定。

考量和限制

在變更預設網路介面設定時，請注意下列考量和限制：

- 您必須在網路介面中設定安全群組，而非在範本的安全群組區段進行設定。您無法在這兩處同時指定安全群組。
- 若您要指定現有網路介面 ID，則僅能啟動一個執行個體。若要這樣做，您必須使用 AWS CLI 或 SDK 來建立 Auto Scaling 群組。在建立群組時，必須指定可用區域，但不須指定子網路 ID。此外，僅在裝置索引為 0 時，您才可以指定現有網路介面。
- 如果指定多個網路介面，則無法自動指派公有 IPv4 地址。您也無法跨網路介面指定重複的裝置索引。主要和次要網路介面都會位於同一子網路中。
- 在執行個體啟動時，會將私有地址自動分配至各網路介面。該地址來自子網路的 CIDR 範圍，執行個體即於該子網路內啟動。如需有關為 VPC 或子網路指定 CIDR 區塊 (或 IP 地址) 的資訊，請參閱《[Amazon VPC 使用者指南](#)》。

變更預設網路介面設定

1. 在 Network settings (網路設定) 下，展開 Advanced network configuration (進階網路組態)。
2. 選擇 Add network interface (新增網路介面)，以設定主要網路介面，請留意以下欄位：

- a. Device index (裝置索引)：保留預設值 0，以將您的變更套用至主要網路介面 (eth0)。
- b. 網路介面 (Network interface)：保留預設值 New interface (新介面)，以便讓 Amazon EC2 Auto Scaling 在執行個體啟動時自動建立新的網路介面。或者，您可以選擇裝置索引為 0 的現有可用網路介面，但這會將 Auto Scaling 群組的執行個體數量限制為一個。
- c. Description (描述)：(選用) 輸入描述性名稱。
- d. Subnet (子網路)：保留預設的 Don't include in launch template (請勿包含於啟動範本中) 設定。

如果 AMI 為網路介面指定子網路，則會導致此錯誤發生。建議解決方法為關閉 Auto Scaling guidance (Auto Scaling 指引)。完成此變更後，便不會再收到錯誤訊息。但是，無論在何處指定子網路，Auto Scaling 群組的子網路設定具有優先順序，且不會被覆寫。

- e. Auto-assign public IP (自動指派公有 IP)：設定裝置索引為 0 的網路介面是否能會收到公有 IPv4 地址。根據預設，預設子網路中的執行個體會收到公有 IPv4 地址，但非預設子網路中的執行個體不會收到。選取 Enable (啟用) 或 Disable (停用) 來覆寫子網路的預設設定。
 - f. Security groups (安全群組)：為網路介面選擇一個或多個安全群組。每個安全群組都必須針對 VPC (Auto Scaling 群組將在其中啟動執行個體) 進行設定。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[適用於 Linux 執行個體的 Amazon EC2 安全群組](#)一節。
 - g. Delete on termination (在終止時刪除)：選擇 Yes (是) 即可在執行個體終止時刪除網路介面，或選擇 No (否) 來保留網路介面。
 - h. Elastic Fabric Adapter：若要支援高效能運算和機器學習使用案例，請將網路介面變更為 Elastic Fabric Adapter 網路介面。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[Elastic Fabric Adapter](#)。
 - i. Network card index (網路卡索引)：選擇 0 即可將主要網路介面連接至裝置索引為 0 的網路卡。如果此選項不可用，請保留預設值，即 Don't include in launch template (請勿包含於啟動範本中)。將網路介面連接至特定網路卡僅適用於受支援的執行個體類型。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[網路卡](#)。
 - j. ENA Express：對於支援 ENA Express 的執行個體類型，選擇啟用以啟用 ENA Express 或停用以停用它。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[在 Linux 執行個體上使用 ENA Express 改善網路效能](#)。
 - k. ENA Express UDP：如果您啟用 ENA Express，您可以選擇將其用於 UDP 流量。選擇啟用以啟用 ENA Express UDP，或選擇停用以停用它。
3. 若要新增次要網路介面，請選擇 Add network interface (新增網路介面)。

修改儲存組態 (主控台)

如果執行個體是從 Amazon EBS 支援的 AMI 或執行個體存放區支援的 AMI 啟動，您可以修改該執行個體的儲存組態。您也可以指定要連接至執行個體的其他 EBS 磁碟區。AMI 包含一個或多個儲存磁碟區，包括根磁碟區 (Volume 1 (AMI Root))。

修改儲存組態

1. 在 Configure storage (設定儲存) 中，修改磁碟區的大小或類型。

如果您指定的磁碟區大小超出磁碟區類型限制，或小於快照大小，則系統會顯示錯誤訊息。為了協助您解決此問題，此訊息會提供該欄位可接受的最小值或最大值。

僅會顯示與 Amazon EBS 支援的 AMI 相關聯的磁碟區。若要顯示從執行個體存放區支援的 AMI 啟動之執行個體儲存組態的相關訊息，請從 instance store volumes (執行個體存放區磁碟區) 區段選擇 Show details (顯示詳細資訊)。

若要指定所有 EBS 磁碟區參數，請在右上角切換至 Advanced (進階) 畫面。

2. 如需進階選項，請展開要修改的磁碟區，然後依照以下步驟設定磁碟區：
 - a. Storage type (儲存體類型)：要與您的執行個體相關聯的磁碟區類型 (EBS 或暫時性)。執行個體存放區 (暫時性) 磁碟區類型僅適用於您選取支援的執行個體類型。如需詳細資訊，請參閱 [《Amazon EBS 使用者指南》中的 Amazon EBS 磁碟區](#) 和 [《Amazon EC2 使用者指南》中的 Amazon EC2 執行個體存放區](#)。 Amazon EC2
 - b. Device name (設備名稱)：從磁碟區可用的設備名稱清單中選取。
 - c. Snapshot (快照)：選取要從中建立磁碟區的快照。您可以在 Snapshot (快照) 欄位中輸入文字，搜尋可用的共享和公有快照。
 - d. Size (GiB) (大小 (GiB))：針對 EBS 磁碟區，您可以指定儲存體大小。如果已選取符合免費方案資格的 AMI 和執行個體，請注意，若要維持在免費方案中，您的總儲存體必須低於 30 GiB。如需詳細資訊，請參閱 [《Amazon EBS 使用者指南》中的 EBS 磁碟區的大小和組態限制](#)。
 - e. Volume type (磁碟區類型)：如果是 EBS 磁碟區，請選擇磁碟區類型。如需詳細資訊，請參閱「Amazon EBS 使用者指南」中的 [Amazon EBS 磁碟區類型](#)。
 - f. IOPS：如果您選取佈建 IOPS SSD (io1 和 io2) 或一般用途 SSD (gp3) 磁碟區類型，則您可輸入磁碟區可支援的每秒輸入/輸出操作數 (IOPS)。對於 io1、io2 和 gp3 磁碟區，這是必要項目。gp2、st1、sc1 或標準磁碟區不支援此功能。

- g. Delete on termination (終止時刪除)：如果是 EBS 磁碟區，選擇 Yes (是) 即可在執行個體終止時刪除磁碟區，或選擇 No (否) 來保留磁碟區。
- h. Encrypted (已加密)：如果執行個體類型支援 EBS 加密，您可以選擇 Yes (是) 以啟用磁碟區的加密。如果您在此區域中預設啟用加密，則會為您啟用預設加密。如需詳細資訊，請參閱《[Amazon EBS 使用者指南](#)》中的 [Amazon EBS 加密](#) 和 [預設啟用 Amazon EBS 加密](#)。

設定此參數的預設效果會隨著磁碟區來源的選擇而改變，如下表所述。在所有情況下，您必須擁有使用指定的許可 AWS KMS key。

加密結果

如果 Encrypted 參數設為...	如果來源磁碟區是...	則預設加密狀態是...	備註
否	新的 (空白) 磁碟區	未加密*	N/A
	您擁有的未加密快照	未加密*	
	您擁有的加密快照	以相同金鑰加密	
	與您共用的未加密快照	未加密*	
	與您共用的加密快照	按預設 KMS 金鑰加密	
是	新磁碟區	按預設 KMS 金鑰加密	若要使用非預設 KMS 金鑰，請指定 KMS 金鑰參數的值。
	您擁有的未加密快照	按預設 KMS 金鑰加密	
	您擁有的加密快照	以相同金鑰加密	
	與您共用的未加密快照	按預設 KMS 金鑰加密	
	與您共用的加密快照	按預設 KMS 金鑰加密	

*若 Encryption by default (預設加密) 已啟用，則所有新建立的磁碟區 (無論 Encrypted (加密) 參數是否已設為 Yes (是)) 都會使用預設 KMS 金鑰加密。若您同時設定 Encrypted (加密) 和 KMS key (KMS 金鑰) 參數，則您可指定非預設 KMS 金鑰。

- i. KMS Key (KMS 金鑰)：如果您為 Encrypted (已加密) 選擇 Yes (是)，則必須選取要用來加密磁碟區的客戶受管金鑰。如果在此區域中預設啟用加密，則會為您選取預設客戶受管金鑰。您可以選取不同的金鑰，或指定您之前已使用 AWS Key Management Service 建立之任何客戶受管金鑰的 ARN。
3. 若要指定其他磁碟區，以連接至透過此啟動範本啟動的執行個體，請選擇 Add new volume (新增磁碟區)。

從現有執行個體建立啟動範本 (主控台)

從現有執行個體建立啟動範本

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在導覽窗格的 Instances (執行個體) 下方，選擇 Instances (執行個體)。
3. 選取執行個體，然後選擇 Actions (動作)、Image and templates (映像和範本)、Create template from instance (從執行個體建立範本)。
4. 提供名稱和描述。
5. 在 Auto Scaling guidance (Auto Scaling 指引) 下，選取核取方塊。
6. 視需要調整任何設定，然後選擇 Create launch template (建立啟動範本)。
7. 若要建立 Auto Scaling 群組，請從確認頁面中選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

相關資源

我們提供幾個 JSON 和 YAML 範本程式碼片段，您可以用來了解如何在 CloudFormation 堆疊範本中宣告啟動範本。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [AWS::EC2::LaunchTemplate](#) 和 [使用 區段建立啟動範本 CloudFormation](#)。

如需啟動範本的詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [從啟動範本啟動執行個體](#)。

限制

- 雖然您可以在啟動範本指定子網路，但如果您只使用啟動範本來建立 Auto Scaling 群組，便不需要這麼做。您無法透過在啟動範本中指定子網路來指定 Auto Scaling 群組的子網路。Auto Scaling 群組的子網路取自 Auto Scaling 群組本身的資源定義。
- 如需使用者定義的網路介面的其他限制，請參閱 [變更預設網路介面設定 \(主控台\)](#)。

使用進階設定建立啟動範本

本主題說明如何從 使用進階設定建立啟動範本 AWS 管理主控台。

使用進階設定建立啟動範本

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在導覽窗格的執行個體下，選擇啟動範本，然後選擇建立啟動範本。
3. 設定啟動範本，如下列主題所述：
 - [必要設定](#)
 - [進階設定](#)
4. 選擇 Create launch template (建立啟動範本)。

必要設定

建立啟動範本時，您必須包含下列必要設定。

啟動範本名稱

輸入描述啟動範本的唯一名稱。

應用程式和作業系統映像 (Amazon Machine Image)

選擇您要使用的 Amazon Machine Image (AMI)。您可以搜尋或瀏覽您要使用的 AMI。為了獲得最佳擴展效率，請選擇已完全設定為使用應用程式碼啟動執行個體的自訂 AMI，並在啟動時需要一些修改。

執行個體類型

選擇與您的 AMI 相容的執行個體類型。如果您計劃使用 Auto Scaling 群組自有資源定義中內嵌的多個執行個體類型，則可以略過將執行個體類型新增至啟動範本。只有在您不打算建立[混合執行個體群組](#)時，才需要執行個體類型。

進階設定

進階設定是選用的。如果您未設定任何進階設定，則不會將特定功能新增至您的執行個體。

展開進階詳細資訊區段以檢視進階設定。下列各節說明為 Auto Scaling 群組建立啟動範本時，應著重的實用進階設定。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[進階詳細資訊](#)。

IAM 執行個體描述檔

執行個體描述檔包含您要使用的 IAM 角色。當您的 Auto Scaling 群組啟動 EC2 執行個體時，相關 IAM 角色中定義的許可會授予執行個體上執行的應用程式。如需詳細資訊，請參閱[在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)。

終止保護

啟用此功能時，可防止使用者使用 Amazon EC2 主控台、CLI 命令和 API 操作終止執行個體。終止保護提供額外的保護，防止意外終止。它不會阻止 Amazon EC2 Auto Scaling 終止執行個體。若要控制 Amazon EC2 Auto Scaling 可以終止哪些執行個體，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。

詳細的 CloudWatch 監控

您可以啟用 EC2 執行個體的詳細監控，以允許它們每隔 1 分鐘將指標資料傳送至 Amazon CloudWatch。根據預設，EC2 執行個體會以 5 分鐘的間隔將指標資料傳送至 CloudWatch。需支付額外費用。如需詳細資訊，請參閱[設定 Auto Scaling 執行個體的監控](#)。

點數規格

Amazon EC2 提供爆量效能執行個體，例如 T2、T3 和 T3a，允許應用程式在需要時爆量超過基準 CPU 效能。根據預設，這些執行個體在 CPU 用量受到調節之前，可能會爆量一段時間。您可以選擇性地啟用無限制模式，讓執行個體視需要爆量超過基準。這可讓應用程式在需要時維持高 CPU 效能。可能會收取額外費用。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[使用 Auto Scaling 群組將爆量效能執行個體啟動為無限制](#)。

置放群組名稱

您可以指定置放群組，並使用叢集或分割區策略來影響執行個體在 AWS 資料中心的實際位置。對於小型 Auto Scaling 群組，您也可以使用分散策略。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[置放群組](#)。

搭配 Auto Scaling 群組使用置放群組時，有一些考量：

- 如果同時在啟動範本和 Auto Scaling 群組中指定置放群組，則以 Auto Scaling 群組的置放群組為優先。
- 在 CloudFormation 中，如果您在啟動範本中定義置放群組，請小心。Amazon EC2 Auto Scaling 會在指定的置放群組中啟動執行個體。不過，如果您搭配 Auto Scaling 群組使用 [UpdatePolicy](#)，CloudFormation 將不會收到來自這些執行個體的訊號（不過未來可能會變更）。

購買選項

您可以選擇請求 Spot 執行個體以 Spot 價格請求 Spot 執行個體，以隨需價格為上限，然後選擇自訂以變更預設 Spot 執行個體設定。針對 Auto Scaling 群組，您必須指定一個不含結束日期 (預設) 的一次性請求。如需詳細資訊，請參閱[為具備容錯能力和靈活性的應用程式請求 Spot 執行個體](#)。此設定在特殊情況下可能很有用，但一般而言，最好保留為未指定，並改為建立混合執行個體群組。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

如果您在啟動範本中指定 Spot 執行個體，便無法建立混合執行個體群組。如果您嘗試使用含有混合執行個體群組要求 Spot 執行個體的啟動範本，您會收到下列錯誤訊息：Incompatible launch template: You cannot use a launch template that is set to request Spot Instances (InstanceMarketOptions) when you configure an Auto Scaling group with a mixed instances policy. Add a different launch template to the group and try again.

Capacity Reservation

容量保留可讓您為特定可用區域中的 Amazon EC2 執行個體保留容量，持續時間不限。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[隨需容量保留](#)。

您可以選擇是否要在 中啟動執行個體：

- 任何開啟的容量保留 (開啟)
- 特定容量保留 (依 ID 設定目標)
- 一組容量預留 (依群組設定目標)

若要以特定容量保留為目標，啟動範本中的執行個體類型必須符合保留的執行個體類型。當您建立 Auto Scaling 群組時，請使用與容量保留相同的可用區域。根據 AWS 區域 您選擇的，您可以選擇改為以容量區塊為目標。如需詳細資訊，請參閱[Capacity Blocks 用於機器學習工作負載](#)。

若要將容量保留群組設為目標，請參閱[使用容量預留在特定可用區域中預留容量](#)。透過以容量保留群組為目標，您可以讓容量分散在多個可用區域，以改善彈性。

租用

Amazon EC2 為 EC2 執行個體的租用提供三個選項：

- 共用 (共用) – 多個 AWS 帳戶 可能會共用相同的實體硬體。這是啟動執行個體時的預設租用選項。
- 專用執行個體 (專用) – 您的執行個體在單一租用戶硬體上執行。沒有其他 AWS 客戶共用相同的實體伺服器。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[專用執行個體](#)。

- 專用主機 (專用主機) – 執行個體會在您專用的實體伺服器上執行。使用專用主機可讓您更輕鬆地將擁有專用硬體需求的授權 (BYOL) 帶入 EC2 並符合合規使用案例。如果您選擇此選項，則必須為租用主機資源群組提供主機資源群組。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[專用主機](#)。

只有在您指定主機資源群組時，才支援專用主機。您無法將特定主機 ID 設為目標，或使用主機置放親和性。

- 如果您嘗試使用指定主機 ID 的啟動範本，您會收到下列錯誤訊息：Incompatible launch template: Tenancy host ID is not supported for Auto Scaling.
- 如果您嘗試使用指定主機置放親和性的啟動範本，您會收到下列錯誤訊息：Incompatible launch template: Auto Scaling does not support host placement affinity.

租用主機資源群組

使用 AWS License Manager，您可以將自己的授權帶到，AWS 並集中管理。主機資源群組是連結至特定 License Manager 授權組態的專用主機群組。主機資源群組可讓您在符合您軟體授權需求的專用主機上輕鬆啟動 EC2 執行個體。您不需要事先手動配置專用主機。它們會視需要自動建立。請注意，當您將 AMI 與授權組態建立關聯時，該 AMI 一次只能與一個主機資源群組建立關聯。如需詳細資訊，請參閱《License Manager 使用者指南》中的在 [託管資源群組 AWS License Manager](#)。

授權組態

使用此設定，您可以為執行個體指定授權組態，而不需將其租用限制為專用主機。授權組態會追蹤部署在執行個體上的軟體授權，以便您可以監控授權用量和合規性。如需詳細資訊，請參閱《License Manager 使用者指南》中的[建立自我管理](#)的授權。

可存取的中繼資料

您可以選擇是否啟用或停用對執行個體中繼資料服務的 HTTP 端點的存取。根據預設，HTTP 端點為啟用狀態。若您選擇停用端點，則會關閉對執行個體中繼資料的存取。您可以指定條件，僅在啟用 HTTP 端點時才需要 IMDSv2。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[設定執行個體中繼資料選項](#)。

中繼資料版本

您可以選擇在請求執行個體中繼資料時要求使用執行個體中繼資料服務第 2 版 (IMDSv2)。若不指定數值，則預設會同時支援 IMDSv1 和 IMDSv2。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[設定執行個體中繼資料選項](#)。

中繼資料字符回應跳轉限制

您可以為中繼資料字符設定允許的網路躍點數量。若未指定數值，則預設值為 1。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[設定執行個體中繼資料選項](#)。

使用者資料

您可以透過將 shell 指令碼或 cloud-init 指令指定為使用者資料，在啟動時自訂並完成執行個體的設定。使用者資料會在執行個體最初啟動時執行，讓您可以在啟動時自動安裝應用程式、相依性或自訂。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[在啟動時在 Linux 執行個體上執行命令](#)。

如果您有大型下載或複雜的指令碼，這會增加執行個體準備好使用所需的時間。在這種情況下，您可能需要設定生命週期掛鉤，以延遲執行個體達到 InService 狀態，直到完全佈建為止。如需將生命週期關聯新增至 Auto Scaling 群組的詳細資訊，請參閱[Amazon EC2 Auto Scaling lifecycle hook](#)。

為具備容錯能力和靈活性的應用程式請求 Spot 執行個體

在啟動範本中，您可以選擇性請求沒有結束日期或持續時間的 Spot 執行個體。與 EC2 隨需價格相比，Amazon EC2 Spot 執行個體是以極低折扣提供的備用容量。如能彈性決定應用程式執行的時機以及應用程式是否可以中斷，則競價型執行個體會是一項具成本效益的選擇。如需建立請求 Spot 執行個體之啟動範本的詳細資訊，請參閱[使用進階設定建立啟動範本](#)。

Important

Spot 執行個體通常用於補充隨需執行個體。針對這種情況，您可以在 Auto Scaling 群組的部分設定中，指定用來啟動 Spot 執行個體的相同設定。在將設定指定為 Auto Scaling 群組的一部分時，您僅能在啟動特定數量的隨需執行個體後請求啟動 Spot 執行個體。隨著群組擴展，您可以繼續啟動隨需執行個體和 Spot 執行個體的組合。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

本主題說明如何藉由在啟動範本中 (而非在 Auto Scaling 群組本身中) 指定設定來僅啟動 Auto Scaling 群組中的 Spot 執行個體。若有 Auto Scaling 群組請求具備[啟動組態](#)的 Spot 執行個體，則本主題中的資訊同樣適用。區別在於啟動組態需要最高價，但對於啟動範本，最高價格是可選的。

當您建立啟動範本，以僅啟動 Spot 執行個體時，請記住以下考量因素：

- **Spot 價格。**您只需為您啟動的 Spot 執行個體支付目前的 Spot 價格。此定價會依據長期的供需趨勢隨時間慢慢改變。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [Spot 執行個體](#) 和 [定價和節省](#)。
- **設定您的最高價。**您可以選擇在啟動範本中包含 Spot 執行個體的每小時最高價。如果最高價超過了目前的 Spot 價格，則 Amazon EC2 Spot 服務會在有可用容量時立即完成您的請求。如果 Spot 執行個體的價格上漲超過您 Auto Scaling 群組執行中執行個體的最高價，則其會終止您的執行個體。

Warning

如果您未收到 Spot 執行個體 (例如當您的最高價太低時)，則您的應用程式可能不會執行。要儘可能長時間利用 Spot 執行個體，請將您的最高價設為接近隨需價格。

- **達成可用區域間的平衡。**若您指定多個可用區域，Amazon EC2 Auto Scaling 會將 Spot 請求分配至這些指定區域。若您在一個可用區域的最高價格過低以至於無法履行請求，Amazon EC2 Auto Scaling 會檢查其他區域是否已履行請求。若是如此，Amazon EC2 Auto Scaling 會取消失敗的請求，並將其重新分配至已履行請求的可用區域。若尚未履行請求的可用區域內價格下跌到足以使未來的請求成功時，則 Amazon EC2 Auto Scaling 會重新平衡所有可用區域。
- **Spot 執行個體終止。**Spot 執行個體可以隨時終止。當 Spot 執行個體的可用性或價格改變時，Amazon EC2 Spot 服務可以終止 Auto Scaling 群組中的 Spot 執行個體。擴展或執行運作狀態檢查時，Amazon EC2 Auto Scaling 也能夠以終止隨需執行個體的方式終止 Spot 執行個體。當執行個體終止時，將會刪除所有儲存。
- **保持所需的容量。**當 Spot 執行個體終止時，Amazon EC2 Auto Scaling 群組會嘗試啟動其他 Spot 執行個體來維持群組的所需容量。如果目前 Spot 價格低於最高價，就會啟動 Spot 執行個體。如果 Spot 執行個體的請求失敗，則會繼續嘗試。
- **變更您的最高價。**要更改您的最高價，請建立新的啟動範本或使用新的最高價更新現有啟動範本，然後將其與 Auto Scaling 群組關聯。只要啟動範本中指定的最高價用於這些執行個體，且高於目前的 Spot 價格，則現有的 Spot 執行個體就會繼續執行。如果您未設定最高價，則預設的最高價是隨需價格。

Capacity Blocks 用於機器學習工作負載

Capacity Blocks 可協助您在未來日期保留熱門的 GPU 執行個體，以支援您的短期機器學習 (ML) 工作負載。

如需 Capacity Blocks 及其運作方式的概觀，請參閱《Amazon EC2 使用者指南》中的 [Capacity Blocks 適用於 ML](#) 的。

若要開始使用 Capacity Blocks，您可以在特定可用區域中建立容量保留。Capacity Blocks 會以單一可用區域中的targeted容量保留的形式交付。當您建立啟動範本時，請指定容量區塊的保留 ID 和執行個體類型。然後，更新您的 Auto Scaling 群組，以使用您建立的啟動範本和容量區塊的可用區域。當您的容量區塊保留開始時，請使用排程擴展來啟動與容量區塊保留相同的執行個體數量。

Important

Capacity Blocks 僅適用於特定 Amazon EC2 執行個體類型和 AWS 區域。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[先決條件](#)。

目錄

- [操作準則](#)
- [在啟動範本中指定容量區塊](#)
- [限制](#)
- [相關資源](#)

操作準則

下列是透過 Auto Scaling 群組使用容量區塊時應遵循的基本操作準則。

- 在容量區塊保留結束時間前 30 分鐘以上，將您的 Auto Scaling 群組縮減為零。Amazon EC2 將在容量區塊結束時間前 30 分鐘終止任何仍在執行的執行個體。
- 我們建議您在適當的保留時間使用排程擴展來向外擴展（新增執行個體）和向內擴展（移除執行個體）。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 排程擴展](#)。
- 視需要新增生命週期關聯，以便在縮減期間對執行個體內的應用程式執行正常關閉程序。Amazon EC2 在容量區塊保留結束前 30 分鐘強制終止執行個體之前，請預留足夠時間讓生命週期動作完成程序。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling lifecycle hook](#)。
- 請確定 Auto Scaling 群組在整個保留期間皆指向正確的啟動範本版本。我們建議您指向特定版本的啟動範本，而不是 \$Default 或 \$Latest 版本。

Note

如果您在保留結束之前讓容量區塊執行個體持續執行，且 Amazon EC2 將其回收，則 Auto Scaling 群組的擴展活動狀態為「taken out of service in response to an EC2

health check that indicated it had been terminated or stopped」，即使刻意在容量區塊結束時回收。同樣地，Amazon EC2 Auto Scaling 會嘗試以與未通過運作狀態檢查的任何執行個體相同的方式取代執行個體。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

在啟動範本中指定容量區塊

若要建立以 Auto Scaling 群組特定容量區塊為目標的啟動範本，請使用下列其中一種方法：

Console

在啟動範本中指定容量區塊 (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在頂端導覽列上，選取您建立容量區塊 AWS 區域的。
3. 在導覽窗格的 Instances (執行個體) 底下，選擇 Launch Templates (啟動範本)。
4. 選擇建立啟動範本，然後建立啟動範本。視需要納入 Amazon Machine Image (AMI) 的 ID、執行個體類型，以及任何其他啟動範本設定。
5. 展開進階詳細資訊區段以檢視進階設定。
6. 針對購買選項，選擇容量區塊。
7. 針對容量保留，選擇依據 ID 的目標，然後針對容量保留 - 依據 ID 的目標，選擇現有容量區塊的容量保留 ID。
8. 當您完成時，請選擇建立啟動範本。

如需使用啟動範本建立 Auto Scaling 群組的說明，請參閱 [使用啟動範本建立 Auto Scaling 群組](#)。

AWS CLI

在啟動範本中指定容量區塊 (AWS CLI)

請使用以下 [create-launch-template](#) 命令，建立指定現有容量區塊保留 ID 的啟動範本。將每個###
#####替換為自己的資訊。

```
aws ec2 create-launch-template --launch-template-name my-template-for-capacity-block \
--version-description AutoScalingVersion1 --region us-east-2 \
```

```
--launch-template-data file://config.json
```

 Tip

如果此命令擲回錯誤，請確定您已在 AWS CLI 本機將 更新為最新版本。

config.json 的內容。

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

以下為範例輸出。

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

您可以使用下列 [describe-launch-template-versions](#) 命令，來驗證容量區塊保留 ID 是否與啟動範本關聯。

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

下列是指定容量保留的啟動範本之範例輸出。

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

限制

- Capacity Blocks 只有在 Auto Scaling 群組具有相容的組態時，才支援。不支援混合執行個體群組和暖集區。
- 您一次只能以一個容量區塊為目標。

相關資源

- 如需使用 P5 執行個體的先決條件和建議，請參閱《Amazon EC2 使用者指南》中的 [P5 執行個體入門](#)。
- Amazon EKS 支援使用 Capacity Blocks來支援 Amazon EKS 叢集上的短期機器學習 (ML) 工作負載。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的[Capacity Blocks適用於 ML](#) 的。
- 您可以Capacity Blocks搭配支援的執行個體類型和區域使用。不過，隨需容量保留可讓您靈活地為其他執行個體類型和區域保留容量。如需示範如何使用隨需容量保留選項的教學課程，請參閱 [使用容量預留在特定可用區域中預留容量](#)。

遷移 Auto Scaling 群組以啟動範本

自 2023 年 1 月 1 日起，啟動組態不再支援新的執行個體類型。這適用於初始區域啟動 AWS 區域 後新增至的任何執行個體類型。此外，您可能可以使用區域中不再支援的執行個體類型建立啟動組態。如需詳細資訊，請參閱[Auto Scaling 啟動組態](#)。

若要將 Auto Scaling 群組從啟動組態遷移至啟動範本，請參閱下列步驟。

Important

確認您具有使用啟動範本所需的許可。如需詳細資訊，請參閱[使用啟動範本的許可](#)。請確定不會意外刪除與啟動組態相關聯的資源，例如安全群組和 IAM 執行個體描述檔。測試復原操作，以確保可以安全地還原變更。

步驟 1：尋找使用啟動組態的 Auto Scaling 群組

若要識別您是否有仍使用啟動組態的 Auto Scaling 群組，請使用 AWS CLI執行下列 [describe-auto-scaling-groups](#) 命令。將 **REGION** 取代為 AWS 區域。

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

以下為範例輸出。

```
[
  {
    "AutoScalingGroupName": "group-1",
```



```

"AutoScalingGroupARN": "arn",
"LaunchConfigurationName": "my-launch-config",
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 2,
"DefaultCooldown": 300,
"AvailabilityZones": [
    "us-west-2a",
    "us-west-2b",
    "us-west-2c"
],
"LoadBalancerNames": [],
"TargetGroupARNs": [],
"HealthCheckType": "EC2",
"HealthCheckGracePeriod": 300,
"Instances": [
    {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchConfigurationName": "my-launch-config",
        "InstanceId": "i-05b4f7d5be44822a6",
        "InstanceType": "t3.micro",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
    },
    {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2b",
        "LaunchConfigurationName": "my-launch-config",
        "InstanceId": "i-0c20ac468fa3049e8",
        "InstanceType": "t3.micro",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
    }
],
"CreatedTime": "2023-03-09T22:15:11.611Z",
"SuspendedProcesses": [],
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
"EnabledMetrics": [],
"Tags": [
    {
        "ResourceId": "group-1",
        "ResourceType": "auto-scaling-group",
        "Key": "environment",

```

```

    "Value": "production",
    "PropagateAtLaunch": true
  },
],
"TerminationPolicies": [
  "Default"
],
"NewInstancesProtectedFromScaleIn": false,
"ServiceLinkedRoleARN": "arn",
  "TrafficSources": []
},

... additional groups ...

]

```

或者，若要移除輸出中含有 Auto Scaling 群組各自啟動組態名稱和標記的 Auto Scaling 群組名稱以外之所有內容，請執行下列命令：

```

aws autoscaling describe-auto-scaling-groups --region REGION \
  --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`].{AutoScalingGroupName:
AutoScalingGroupName, LaunchConfigurationName: LaunchConfigurationName, Tags: Tags}'

```

下面顯示了範例輸出。

```

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

  ... additional groups ...
]

```

]

如需篩選的詳細資訊，請參閱AWS Command Line Interface 《使用者指南》中的[篩選 AWS CLI 輸出](#)。

步驟 2：將啟動組態複製到啟動範本

您可以依照下列程序將啟動組態複製到啟動範本。然後，您可以將其新增到 Auto Scaling 群組。

複製多個啟動組態會產生相同名稱的啟動範本。若要在複製期間變更指定給啟動範本的名稱，您必須逐一複製啟動組態。

Note

複製功能僅能從主控台使用。

將啟動組態複製到啟動範本 (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在左側導覽窗格的 Auto Scaling 下，選擇 Auto Scaling 群組。
3. 在頁面頂端附近選擇啟動組態。出現確認提示時，請選擇檢視啟動組態，以確認您要檢視啟動組態頁面。
4. 選取要複製的啟動組態，然後選擇 Copy to launch template, Copy selected (複製至啟動範本，複製選取部分)。如此會以相同於您選取的啟動組態名稱和選項來設定新的啟動範本。
5. 對於 New launch template name (新的啟動範本名稱)，您可以使用啟動組態的名稱 (預設) 或輸入新的名稱。啟動範本名稱必須是唯一的。
6. (選用) 選取使用新範本建立 Auto Scaling 群組。

您可以略過此步驟，以完成啟動組態的複製作業。您不需要建立新的 Auto Scaling 群組。

7. 請選擇 Copy (複製)。

將所有啟動組態複製到啟動範本 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在導覽窗格的 Auto Scaling 下方，選擇 Launch Configurations (啟動組態)。

3. 選擇 Copy to launch template, Copy all (複製到啟動範本，複製全部)。這會將目前區域中的每個啟動組態複製到具有相同名稱和選項的新啟動範本。
4. 請選擇 Copy (複製)。

步驟 3：更新 Auto Scaling 群組以使用啟動範本

建立啟動範本後，您便可以將範本新增至 Auto Scaling 群組。

更新 Auto Scaling 群組以使用啟動範本 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格，顯示所選群組的相關資訊。

3. 在 Details (詳細資訊) 索引標籤上，請依序選擇 Launch configuration (啟動組態) 和 Edit (編輯)。
4. 選擇 Switch to launch template (切換至啟動範本)。
5. 針對 Launch Template (啟動範本)，選取您的啟動範本。
6. 針對 Version (版本)，請視需要選取啟動範本版本。建立許多啟動範本版本後，您可以在擴展時選擇 Auto Scaling 群組是否使用預設或最新版本的啟動範本。
7. 選擇更新。

更新 Auto Scaling 群組以使用啟動範本 (AWS CLI)

下列 [update-auto-scaling-group](#) 命令會更新指定的 Auto Scaling 群組，以便使用指定啟動範本的初始版本。

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

如需使用 CLI 命令更新 Auto Scaling 群組以使用啟動範本的範例，請參閱 [更新 Auto Scaling 群組以使用啟動範本](#)。

步驟 4：取代您的執行個體

使用啟動範本取代啟動組態後，任何新執行個體都會使用新啟動範本。現有的執行個體不受影響。

若要更新現有執行個體，您可以啟動執行個體重新整理來取代 Auto Scaling 群組中的執行個體，而不是一次手動取代一些執行個體。如需詳細資訊，請參閱[使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。如果群組較大，執行個體重新整理可能特別有用。

或者，您也可以允許自動擴展功能，根據群組的[終止政策](#)逐漸將現有的執行個體取代為新執行個體，或是您也可以終止這些執行個體。手動終止會強制 Auto Scaling 群組啟動新執行個體，以維持群組的所需容量。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[終止執行個體](#)。

其他資訊

如需詳細資訊，請參閱 AWS 運算部落格上的 [Amazon EC2 Auto Scaling 將不再將對新 EC2 功能的支援新增至啟動組態](#)。

如需帶您了解如何將 AWS CloudFormation 堆疊從啟動組態遷移至啟動範本的主題，請參閱 [遷移 AWS CloudFormation 堆疊以啟動範本](#)。

遷移 AWS CloudFormation 堆疊以啟動範本

您可以將現有的 CloudFormation 堆疊範本從啟動組態遷移到啟動範本。為此，請將啟動範本直接新增至現有的堆疊範本，然後將啟動範本與堆疊範本中的 Auto Scaling 群組建立關聯。然後，使用修改過的範本更新堆疊。

遷移到啟動範本時，本主題會提供將 CloudFormation 堆疊範本中的啟動組態重寫為啟動範本的指示，以節省您的時間。如需將啟動組態遷移到啟動範本的詳細資訊，請參閱 [遷移 Auto Scaling 群組以啟動範本](#)。

主題

- [尋找使用啟動組態的 Auto Scaling 群組](#)
- [更新堆疊以使用啟動範本](#)
- [了解更新堆疊資源的行為](#)
- [追蹤遷移](#)
- [啟動組態對應參考](#)

尋找使用啟動組態的 Auto Scaling 群組

尋找使用啟動組態的 Auto Scaling 群組

- 使用下列 [describe-auto-scaling-groups](#) 命令，列出在指定區域中使用啟動組態的 Auto Scaling 群組的名稱。包含 `--filters` 選項，以將結果縮小到與 CloudFormation 堆疊關聯的群組 (透過 `aws:cloudformation:stack-name` 標籤金鑰進行篩選)。

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

下面顯示了範例輸出。

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

您可以找到其他有用的 AWS CLI 命令，以尋找 Auto Scaling 群組來遷移和篩選 中的輸出[遷移 Auto Scaling 群組以啟動範本](#)。

Important

如果您的堆疊資源名稱 AWSEB 中有，這表示它們是透過 建立 AWS Elastic Beanstalk 的。在這種情況下，您必須更新 Beanstalk 環境以指示 Elastic Beanstalk 移除啟動組態並將其替換為啟動範本。

更新堆疊以使用啟動範本

請依照本章節中的步驟，執行下列操作：

- 使用同等的啟動範本屬性，將啟動組態重寫為啟動範本。
- 將新的啟動範本與 Auto Scaling 群組關聯。
- 部署這些更新。

修改堆疊範本並更新堆疊

1. 請遵循與《AWS CloudFormation 使用者指南》中所述的[修改堆疊範本](#)相同的一般程序。
2. 將啟動組態重寫為啟動範本。請參閱下列範例：

範例：簡單的啟動組態

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

範例：啟動範本對等項目

```
---
Resources:
  myLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: ami-02354e95b3example
        InstanceType: t3.micro
        SecurityGroupIds:
          - Ref! EC2SecurityGroup
```



```

KeyName: MyKeyPair
BlockDeviceMappings:
  - DeviceName: /dev/xvda
    Ebs:
      VolumeSize: 150
      DeleteOnTermination: true
UserData:
  Fn::Base64: !Sub |
    #!/bin/bash -x
    yum install -y aws-cfn-bootstrap
    /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
    myASG --region ${AWS::Region}

```

如需 Amazon EC2 支援的所有屬性的參考資訊，請參閱《AWS CloudFormation 使用者指南》中的 [AWS::EC2::LaunchTemplate](#)。

請注意啟動範本如何包含值為 `!Sub ${AWS::StackName}-launch-template` 的 `LaunchTemplateName` 屬性。如果您想要啟動範本的名稱包含堆疊名稱，則需執行此動作。

3. 如果啟動組態中存在 **IamInstanceProfile** 屬性，則必須將其轉換成一個結構，並指定執行個體設定檔的名稱或 ARN。如需範例，請參閱 [AWS::EC2::LaunchTemplate](#)。
4. 如果啟動組態中存在 **AssociatePublicIpAddress**、**InstanceMonitoring** 或 **PlacementTenancy** 屬性，則必須將其轉換成一個結構。如需範例，請參閱 [AWS::EC2::LaunchTemplate](#)。

當用於 Auto Scaling 群組的子網路上的 `MapPublicIpOnLaunch` 屬性值與啟動組態中的 `AssociatePublicIpAddress` 屬性值相符時，則屬於例外情況。在這種情況下，可以忽略 `AssociatePublicIpAddress` 屬性。`AssociatePublicIpAddress` 屬性僅用於覆寫 `MapPublicIpOnLaunch` 屬性，以變更執行個體在啟動時是否接收公用 IPv4 位址。

5. 您可以將安全群組從 **SecurityGroups** 屬性複製到啟動範本中的兩個位置之一。一般來說，您可以將安全群組複製到 `SecurityGroupIds` 屬性。但是，如果您在啟動範本中建立 `NetworkInterfaces` 結構來指定 `AssociatePublicIpAddress` 屬性，則必須將安全群組複製到網路介面的 `Groups` 屬性。
6. 如果啟動組態中存在任何將 **NoDevice** 設定為 `true` 的 `BlockDeviceMapping` 結構，則必須在啟動範本中為 `NoDevice` 指定空白字串，以讓 Amazon EC2 省略該裝置。
7. 如果啟動配置中存在 **SpotPrice** 屬性，建議您將其從啟動模板中省略。您的 Spot 執行個體將以目前的 Spot 價格啟動。價格絕不會超過隨需價格。

若要請求 Spot 執行個體，您會有兩個互斥的選項：

- 第一個選項是使用啟動範本中的 `InstanceMarketOptions` 結構 (不建議使用)。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 [AWS::EC2::LaunchTemplate InstanceMarketOptions](#)。
 - 另一個選項是將 `MixedInstancesPolicy` 結構新增至您的 Auto Scaling 群組。這樣做會為您發出請求的方式提供更多選項。啟動範本中的 Spot 執行個體請求不支援每個 Auto Scaling 群組選取一個以上的執行個體類型。但是混合執行個體政策則支援每個 Auto Scaling 群組選取一個以上的執行個體類型。Spot 執行個體請求受益於有多種執行個體類型可供選擇。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 [AWS::AutoScaling::AutoScalingGroup MixedInstancesPolicy](#)。
8. 將 `LaunchConfigurationName` 屬性從 [AWS::AutoScaling::AutoScalingGroup](#) 資源中移除。在該位置新增啟動範本。

在下列範例中，[參考](#)內部函數會取得邏輯 ID 為 `myLaunchTemplate` 的 [AWS::EC2::LaunchTemplate](#) 資源的 ID。[GetATT](#) 函數可取得 `Version` 屬性之啟動範本的最新版本號碼 (例如 1)。

範例：沒有混合執行個體政策

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

範例：有混合執行個體政策

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

```
Version: !GetAtt myLaunchTemplate.LatestVersionNumber
```

```
...
```

若需要 Amazon EC2 Auto Scaling 支援的所有屬性的參考資訊，請參閱《AWS CloudFormation 使用者指南》中的 [AWS::AutoScaling::AutoScalingGroup](#)。

9. 當您準備好部署這些更新時，請遵循 CloudFormation 程序，使用修改過的堆疊範本更新堆疊。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 [修改堆疊範本](#)。

了解更新堆疊資源的行為

CloudFormation 會比較您提供的更新範本與您在舊版堆疊範本中描述的資源組態之間的變更，藉此更新堆疊資源。尚未變更的資源組態在更新程序期間不會受到影響。

CloudFormation 支援 Auto Scaling 群組的 [UpdatePolicy](#) 屬性。在更新期間，如果 UpdatePolicy 設定為 AutoScalingRollingUpdate，CloudFormation 會在您執行此程序中的步驟後取代 InService 執行個體。如果 UpdatePolicy 設定為 AutoScalingReplacingUpdate，CloudFormation 會取代 Auto Scaling 群組及其暖集區 (如果有的話)。

如果您未指定 Auto Scaling 群組的 UpdatePolicy 屬性，則會檢查啟動範本的正确性，但 CloudFormation 不會在 Auto Scaling 群組的執行個體中部署任何變更。所有新的執行個體均將使用您的啟動範本，但現有的執行個體會繼續使用其原先啟動的組態來執行 (儘管啟動組態已不存在)。當您變更購買選項時，例如新增混合執行個體政策時，則屬例外情況。在這種情況下，Auto Scaling 群組會逐漸以新的執行個體取代現有的執行個體，以符合新的購買選項。

如果您必須復原變更，才能從啟動組態移至啟動範本，請務必測試復原操作。

追蹤遷移

追蹤遷移

1. 在 [CloudFormation 主控台](#) 選取更新過的堆疊，接著選擇 Events (事件) 標籤來檢視堆疊事件。
2. 如需更新事件清單以包含最新的事件，則請選擇 CloudFormation 主控台中的重新整理按鈕。
3. 當您的堆疊更新時，您會發現每個資源更新了多個事件。如果您在狀態原因欄中看到例外狀況，表示嘗試建立啟動範本時發生了問題，請參閱 [Amazon EC2 Auto Scaling 的故障診斷：啟動範本](#) 了解潛在原因。
4. (可選性) 根據您對 UpdatePolicy 屬性的使用情況，您可以從 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#) 監控 Auto Scaling 群組的進度。選取 Auto Scaling 群組。在 Activity (活動) 索引

標籤的 Activity history (活動歷史記錄) 中，Status (狀態) 欄位會顯示 Auto Scaling 群組是否已成功啟動或終止執行個體，或擴展活動是否仍在進行中。

- 堆疊更新完成時，CloudFormation 會發布 UPDATE_COMPLETE 堆疊事件。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的[監控堆疊更新的進度](#)。
- 堆疊更新完成後，打開 Amazon EC2 主控台的[啟動範本頁面](#)和[啟動組態頁面](#)。您會發現新的啟動範本已建立，並且啟動組態已刪除。

啟動組態對應參考

為參考起見，下表列出了 [AWS::AutoScaling::LaunchConfiguration](#) 資源中的所有頂層屬性以及它們在 [AWS::EC2::LaunchTemplate](#) 資源中的對等屬性。

啟動組態來源屬性	啟動範本目標屬性
AssociatePublicIpAddress	NetworkInterfaces.AssociatePublicIpAddress
BlockDeviceMappings	BlockDeviceMappings
ClassicLinkVPCId	不適用 ¹
ClassicLinkVPCSecurityGroups	不適用 ¹
EbsOptimized	EbsOptimized
IamInstanceProfile	IamInstanceProfile.Arn 或 IamInstanceProfile.Name 其中之一
ImageId	ImageId
InstanceId	InstanceId
InstanceMonitoring	Monitoring.Enabled
InstanceType	InstanceType
KernelId	KernelId
KeyName	KeyName

啟動組態來源屬性	啟動範本目標屬性
LaunchConfigurationName	LaunchTemplateName
MetadataOptions	MetadataOptions
PlacementTenancy	Placement.Tenancy
RamDiskId	RamDiskId
SecurityGroups	SecurityGroupIds 或 NetworkInterfaces.Groups 其中之一
SpotPrice	InstanceMarketOptions.SpotOptions.MaxPrice
UserData	UserData

1 ClassicLinkVPCId和 ClassicLinkVPCSecurityGroups 屬性無法用於啟動範本，因為 EC2-Classic 不再可用。

使用 建立和管理啟動範本的範例 AWS CLI

您可以透過 AWS 管理主控台、AWS Command Line Interface (AWS CLI) 或 SDKs 建立和管理啟動範本。本節顯示從 建立和管理 Amazon EC2 Auto Scaling 啟動範本的範例 AWS CLI。

目錄

- [範例使用方式](#)
- [建立基本的啟動範本](#)
- [指定在啟動時標記執行個體的標籤](#)
- [指定要傳遞給執行個體的 IAM 角色](#)
- [指派公有 IP 地址](#)
- [指定啟動時設定執行個體的使用者資料指令碼](#)
- [指定區塊型設備映射](#)
- [指定專用主機以從外部廠商取得軟體授權](#)
- [指定現有網路介面](#)

- [建立多個網路介面](#)
- [管理啟動範本](#)
- [更新 Auto Scaling 群組以使用啟動範本](#)

範例使用方式

```
{
  "LaunchTemplateName": "my-template-for-auto-scaling",
  "VersionDescription": "test description",
  "LaunchTemplateData": {
    "ImageId": "ami-04d5cc9b88example",
    "InstanceType": "t2.micro",
    "SecurityGroupIds": [
      "sg-903004f88example"
    ],
    "KeyName": "MyKeyPair",
    "Monitoring": {
      "Enabled": true
    },
    "Placement": {
      "Tenancy": "dedicated"
    },
    "CreditSpecification": {
      "CpuCredits": "unlimited"
    },
    "MetadataOptions": {
      "HttpTokens": "required",
      "HttpPutResponseHopLimit": 1,
      "HttpEndpoint": "enabled"
    }
  }
}
```

建立基本的啟動範本

若要建立基本的啟動範本，請按如下所示使用 [create-launch-template](#) 命令，且必須進行這些修改：

- 以從中啟動執行個體的 AMI 的 ID 取代 `ami-04d5cc9b88example`。
- 以與您所指定的 AMI 相容的執行個體取代 `t2.micro`。

此範例會建立名為 *my-template-for-auto-scaling* 的啟動範本。如果此啟動範本建立的執行個體在預設 VPC 中啟動，則依預設，它們會接收公有 IP 地址。如果執行個體是在非預設的 VPC 中啟動，則其預設不會接收公有 IP 位址。

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

如需有關引用 JSON 格式參數的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[在 AWS CLI 中搭配字串使用引號](#)。

您也可以組態檔案中指定 JSON 格式參數。

下列範例會建立基本的啟動範本，其會參考啟動範本參數值的組態檔案。

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

config.json 的內容：

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

指定在啟動時標記執行個體的標籤

下列範例會在啟動時將標籤 (例如：purpose=webserver) 新增至執行個體。

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


Note

如果在啟動範本中指定執行個體標籤，接著選擇將 Auto Scaling 群組的標籤傳播到其執行個體，則會合併所有標籤。如果為啟動範本中的標籤和 Auto Scaling 群組中的標籤指定了相同的標籤鍵，則以群組中的標籤值為優先。

指定要傳遞給執行個體的 IAM 角色

下列範例會指定與 IAM 角色相關聯的執行個體描述檔名稱，以便在啟動時傳遞給執行個體。如需詳細資訊，請參閱 [在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)。

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

指派公有 IP 地址

下列 [create-launch-template](#) 範例會設定啟動範本，將公有地址指派給在非預設 VPC 中啟動的執行個體。

Note

在指定網路介面時，請為 Groups，即對應至 Auto Scaling 群組在其中啟動執行個體的 VPC 安全群組指定數值。將 VPC 子網路指定為 Auto Scaling 群組的屬性。

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

指定啟動時設定執行個體的使用者資料指令碼

下列範例會將使用者資料指令碼指定為在啟動時設定執行個體的 base64 編碼字串。 [create-launch-template](#) 命令需要 base64 編碼的使用者資料。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data
'{"UserData":"IyEvYmluL2Jhc...", "ImageId":"ami-04d5cc9b88example", "InstanceType":"t2.micro"}'
```

指定區塊型設備映射

下列 [create-launch-template](#) 範例會建立具備區塊型設備映射的啟動範本：一個 22 GB 的 EBS 磁碟區，並將其映射至 `/dev/xvdcz`。`/dev/xvdcz` 磁碟區使用一般用途 SSD (gp2) 磁碟區類型，並會在與其相連接的執行個體終止時刪除。

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

指定專用主機以從外部廠商取得軟體授權

若您指定主機租用，則可以指定主機資源群組和 License Manager 授權組態，以便從外部廠商取得合格的軟體授權。然後，您可以藉由下列 [create-launch-template](#) 命令，在 EC2 執行個體上使用授權。

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

指定現有網路介面

下列 [create-launch-template](#) 範例會將主要網路介面設定為使用現有網路介面。

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

建立多個網路介面

下列 [create-launch-template](#) 範例會新增次要網路介面。主要網路介面的裝置索引為 0，而次要網路介面的裝置索引為 1。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
  --launch-template-data '{"NetworkInterfaces":[{"DeviceIndex":0,"Groups":  
["sg-903004f8example"],"DeleteOnTermination":true},{"DeviceIndex":1,"Groups":  
["sg-903004f8example"],"DeleteOnTermination":true],"ImageId":"ami-04d5cc9b88example","InstanceType":t3.xlarge}'
```

如果使用支援多張網路卡和 Elastic Fabric Adapter (EFA) 的執行個體類型，您可以將次要介面新增至次要網路卡，再使用下列 [create-launch-template](#) 命令啟用 EFA。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[將 EFA 新增至啟動範本](#)。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
  --launch-template-data '{"NetworkInterfaces":  
[{"NetworkCardIndex":0,"DeviceIndex":0,"Groups":  
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true},  
{"NetworkCardIndex":1,"DeviceIndex":1,"Groups":  
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true],"ImageId":"ami-09d955f2example","InstanceType":p4d.24xlarge}'
```

Warning

p4d.24xlarge 執行個體類型所產生的成本高於本節中的其他範例。如需有關 P4d 執行個體定價的詳細資訊，請參閱 [Amazon EC2 P4d 執行個體定價](#)。

Note

將相同子網路中的多個網路介面連接至執行個體，會導致非對稱路由，這種情況特別容易發生在使用非 Amazon Linux 變體的執行個體上。如果需要這種類型的組態，您必須在作業系統內設定次要網路介面。如需範例，請參閱 AWS 知識中心的[如何讓次要網路界面在 Ubuntu EC2 執行個體中運作？](#)。

管理啟動範本

AWS CLI 包含數個其他命令，可協助您管理啟動範本。

目錄

- [列出並描述啟動範本](#)
- [建立啟動範本版本](#)
- [刪除啟動範本版本](#)
- [刪除啟動範本](#)

列出並描述啟動範本

您可以使用兩個 AWS CLI 命令來取得啟動範本的相關資訊：[describe-launch-templates](#) 和 [describe-launch-template-versions](#)。

[describe-launch-templates](#) 命令能取得您已建立的任何啟動範本的清單。您可以使用選項來篩選啟動範本名稱的結果、建立時間、標籤鍵或標籤鍵值對。此命令會傳回任何啟動範本的摘要資訊，包括啟動範本識別符、最新版本和預設版本。

下列範例提供所指定啟動範本的摘要。

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

以下是回應範例。

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

若您沒有使用 `--launch-template-names` 選項將輸出限制為一個啟動範本，則會傳回關於所有啟動範本的資訊。

下列 [describe-launch-template-versions](#) 命令提供描述指定啟動範本之版本的資訊。

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

以下是回應範例。

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

```
        "CreateTime": "2020-02-28T19:52:27.000Z"  
      }  
    ]  
  }  
}
```

建立啟動範本版本

下列 [create-launch-template-version](#) 命令，會建立以第 1 版啟動範本為基礎的新啟動範本版本，並指定不同的 AMI ID。

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

若要設定啟動範本的預設版本，請使用 [modify-launch-template](#) 命令。

刪除啟動範本版本

下列 [delete-launch-template-versions](#) 命令會刪除指定的啟動範本版本。

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

刪除啟動範本

若不再需要啟動範本，即可使用 [delete-launch-template](#) 命令將其刪除。刪除啟動範本會刪除它的所有版本。

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

更新 Auto Scaling 群組以使用啟動範本

您可以使用 [update-auto-scaling-group](#) 命令將啟動範本新增至現有 Auto Scaling 群組。

更新 Auto Scaling 群組以使用最新版本的啟動範本

下列 [update-auto-scaling-group](#) 命令會更新指定的 Auto Scaling 群組，以便使用指定啟動範本的最新版本。

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

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

更新 Auto Scaling 群組以使用特定版本的啟動範本

下列 [update-auto-scaling-group](#) 命令會更新指定的 Auto Scaling 群組，以便使用指定啟動範本的特定版本。

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

在啟動範本中使用 AWS Systems Manager 參數而非 AMI IDs

本節說明如何建立啟動範本，指定參考 AWS Systems Manager Amazon Machine Image (AMI) ID 的參數。您可以使用存放在相同中的參數 AWS 帳戶、從另一個共用的參數 AWS 帳戶，或由維護的公有 AMI 的公有參數 AWS。

透過 Systems Manager 參數，您可以更新 Auto Scaling 群組以使用新的 AMI ID，而不需要在每次 AMI ID 變更時建立新啟動範本或啟動範本的新版本。在透過最新作業系統或軟體更新進行 AMI 更新等情況下，這些 ID 可能會定期變更。

您可以使用參數[存放區功能來建立、更新或刪除自己的 Systems Manager 參數 AWS Systems Manager](#)。您必須先建立 Systems Manager 參數，才能在啟動範本中使用該參數。若要開始使用，請建立具有資料類型 `aws:ec2:image` 的參數，並為其值輸入 AMI ID。AMI ID 具有 `ami-<identifier>` 格式，例如 `ami-123example456`。AMI ID 是否正確將取決於執行個體類型以及啟動 Auto Scaling 群組的 AWS 區域。

如需為 AMI ID 建立有效參數的詳細資訊，請參閱[建立 Systems Manager 參數](#)。

建立指定 AMI 參數的啟動範本

若要建立指定 AMI 參數的啟動範本，請使用下列其中一種方法：

Console

使用 AWS Systems Manager 參數建立啟動範本

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在導覽窗格中，選擇 Launch Templates (啟動範本) 接著選擇 Create launch template (建立啟動範本)。

3. 針對 Launch template name (啟動範本名稱)，輸入啟動範本的描述性名稱。
4. 在 Application and OS Images (Amazon Machine Image) (應用程式和作業系統映像 (Amazon Machine Image)) 下，選擇 Browse more AMIs (瀏覽更多 AMI)。
5. 選擇搜尋列右側的箭頭按鈕，然後選擇指定自訂值/Systems Manager 參數。
6. 在指定自訂值或 Systems Manager 參數對話方塊中，執行下列操作：

- a. 對於 AMI ID 或 Systems Manager 參數字串，請使用下列其中一種格式輸入 Systems Manager 參數名稱：

若要參考公用參數：

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

若要參考存放在相同帳戶中的參數：

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

若要參考從另一個 AWS 帳戶共用的參數：

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

- b. 選擇儲存。

7. 視需要設定任何其他啟動範本設定，然後選擇建立啟動範本。如需詳細資訊，請參閱 [Auto Scaling 群組建立啟動範本](#)。

AWS CLI

若要建立指定 Systems Manager 參數的啟動範本，您可以使用下列其中一個範例命令。將每個 **###** **#####** 替換為自己的資訊。

範例：建立指定 AWS 擁有的公有參數的啟動範本

使用下列語法：**resolve:ssm:*public-parameter***，其中 **resolve:ssm** 是標準前綴，而 ***public-parameter*** 是公有參數的路徑和名稱。

在此範例中，啟動範本使用 AWS 提供的公有參數，在為您的設定檔設定的 中使用最新的 Amazon Linux 2 AMI AWS 區域 啟動執行個體。

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

config.json 的內容：

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

以下是回應範例。

```
{
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-089c023a30example",
    "LaunchTemplateName": "my-template-for-auto-scaling",
    "CreateTime": "2022-12-28T19:52:27.000Z",
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "DefaultVersionNumber": 1,
    "LatestVersionNumber": 1
  }
}
```

範例：建立啟動範本，指定存放在相同帳戶中的參數

使用下列語法：`resolve:ssm:parameter-name`，其中 `resolve:ssm` 是標準字首，而 *parameter-name* 是 Systems Manager 參數名稱。

下列範例會建立從名為 *golden-ami* 的現有 Systems Manager 參數取得 AMI ID 的啟動範本。

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

config.json 的內容：

```
{
```

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

如果未指定，參數的預設版本即為最新版本。

下列範例會參考 *golden-ami* 參數的特定版本。此範例使用 *golden-ami* 參數的第 3 版，但您可以使用任何有效的版本編號。

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

下列類似範例會參考映射到 *golden-ami* 參數特定版本的參數標籤 *prod*。

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

以下為範例輸出。

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

範例：建立啟動範本，指定從另一個 共用的參數 AWS 帳戶

使用以下語法：`resolve:ssm:parameter-ARN`，其中 `resolve:ssm` 是標準字首，而 *parameter-ARN* 是 Systems Manager 參數的 ARN。

下列範例會建立啟動範本，從具有 ARN 的現有 Systems Manager 參數取得 AMI ID *arn:aws:ssm:us-east-2:123456789012:parameter/MyParameter*。

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

config.json 的內容：

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

如果未指定，參數的預設版本即為最新版本。

下列範例會參考 *MyParameter* 參數的特定版本。此範例使用 *MyParameter* 參數的第 3 版，但您可以使用任何有效的版本編號。

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

下列類似範例會參考映射到 *MyParameter* 參數特定版本的參數標籤 *prod*。

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

以下是回應範例。

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

```
}  
}
```

若要從啟動範本中的參數存放區指定參數，您必須具有指定參數的 `ssm:GetParameters` 許可。使用啟動範本的任何人也需要 `ssm:GetParameters` 許可，才能驗證參數值。如需詳細資訊，請參閱AWS Systems Manager 《使用者指南》中的[使用 IAM 政策限制對 Systems Manager 參數的存取](#)。

確認啟動範本取得正確的 AMI ID

使用 [describe-launch-template-versions](#) 命令，並包含將參數解析為實際 AMI ID `--resolve-alias` 的選項。

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

此範例會傳回 ImageId 的 AMI ID。使用此啟動範本啟動執行個體時，AMI ID 會解析為 `ami-0ac394d6a3example`。

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

相關資源

如需在啟動範本中指定 Systems Manager 參數的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[使用 Systems Manager 參數而非 AMI ID](#)。

如需關於使用 Systems Manager 參數的詳細資訊，請參閱 Systems Manager 文件中的下列參考資料。

- 若要建立參數版本和標籤，請參閱[使用參數版本](#)和[使用參數標籤](#)。
- 如需如何查詢 Amazon EC2 所支援 AMI 公用參數的詳細資訊，請參閱[呼叫 AMI 公用參數](#)。
- 如需與其他 AWS 帳戶或透過 共用參數的資訊 AWS Organizations，請參閱[使用共用參數](#)。
- 如需監控參數是否已成功建立的相關資訊，請參閱「Amazon Machine Image ID 的原生參數支援」<https://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-ec2-aliases.html>。

限制

使用 Systems Manager 參數時，請注意下列限制：

- Amazon EC2 Auto Scaling 僅支援指定 AMI ID 作為參數。
- 不支援使用指定 Systems Manager 參數的啟動範本，使用[屬性型執行個體類型選取](#)來建立或更新[混合執行個體群組](#)。
- 如果您的 Auto Scaling 群組使用指定 Systems Manager 參數的啟動範本，您將無法以所需的組態或使用略過比對來啟動執行個體重新整理。
- 如果您的 Auto Scaling 群組使用指定 Systems Manager 參數的啟動範本，則不支援暖集區。
- 每次進行呼叫以建立或更新 Auto Scaling 群組時，Amazon EC2 Auto Scaling 都會解析啟動範本中的 Systems Manager 參數。如果您使用進階參數或較高的輸送量限制，頻繁呼叫 Parameter Store (亦即 GetParameters 操作) 可能會增加 Systems Manager 的成本，因為每個 Parameter Store API 互動都會產生費用。如需詳細資訊，請參閱[AWS Systems Manager 定價](#)。

Auto Scaling 啟動組態

Important

限制:

- 自 2023 年 1 月 1 日起，啟動組態不再支援新的 Amazon EC2 執行個體類型。這包括支援在初始區域啟動 AWS 區域 後新增至 的任何執行個體類型。
- 在 2023 年 6 月 1 日當天或之後建立的帳戶無法使用 主控台 建立新的啟動組態。
- 2024 年 10 月 1 日當天或之後建立的帳戶無法使用任何方法 (主控台 AWS CLI、API 或 CloudFormation) 建立新的啟動組態。

遷移以啟動範本，以確保您現在或未來不需要建立新的啟動組態。如需將 Auto Scaling 群組遷移至啟動範本的資訊，請參閱 [遷移 Auto Scaling 群組以啟動範本](#)。

Note

您可能可以使用 區域中不再支援的執行個體類型來建立啟動組態。我們建議您遷移至啟動範本。

對於那些尚未從啟動組態遷移至啟動範本的客戶，我們會提供啟動組態的資訊。啟動組態是 Auto Scaling 群組用於啟動 EC2 執行個體的執行個體組態範本。當您建立啟動組態時，您可指定執行個體的資訊。包括 Amazon Machine Image (AMI) 的 ID、執行個體類型、金鑰對，一或多個安全群組，以及區塊型設備映射。如果您之前已啟動 EC2 執行個體，您會指定相同的資訊來啟動執行個體。

您能夠以多個 Auto Scaling 群組指定啟動組態。不過，您一次僅能為一個 Auto Scaling 群組指定一個啟動組態，而且建立啟動組態後便無法再修改。若要變更 Auto Scaling 群組的啟動組態，您必須建立啟動組態，然後使用它來更新 Auto Scaling 群組。

目錄

- [建立啟動組態](#)
- [變更 Auto Scaling 群組的啟動組態](#)

建立啟動組態

Important

限制:

- 自 2023 年 1 月 1 日起，啟動組態不再支援新的 Amazon EC2 執行個體類型。這包括支援在初始區域啟動 AWS 區域 後新增至 的任何執行個體類型。
- 在 2023 年 6 月 1 日當天或之後建立的帳戶無法使用 主控台 建立新的啟動組態。
- 2024 年 10 月 1 日當天或之後建立的帳戶無法使用任何方法（主控台 AWS CLI、API 或 CloudFormation）建立新的啟動組態。

遷移以啟動範本，以確保您現在或未來不需要建立新的啟動組態。如需將 Auto Scaling 群組遷移至啟動範本的資訊，請參閱 [遷移 Auto Scaling 群組以啟動範本](#)。

Note

您可能可以使用 區域中不再支援的執行個體類型來建立啟動組態。我們建議您遷移至啟動範本。

本主題說明如何建立啟動組態。對於那些尚未從啟動組態遷移至啟動範本的客戶，我們會提供啟動組態的資訊。

建立啟動組態之後，您就無法修改它。您必須改為建立新的啟動組態。

若要將新的啟動組態與現有的 Auto Scaling 群組建立關聯，請參閱 [變更 Auto Scaling 群組的啟動組態](#)。若要建立新的 Auto Scaling 群組，請參閱 [使用啟動組態建立 Auto Scaling 群組](#)。

目錄

- [建立啟動組態](#)
- [設定執行個體中繼資料選項](#)
- [使用 EC2 執行個體建立啟動組態](#)

建立啟動組態

建立啟動組態 (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在頂端導覽列上，選取您的 AWS 區域。
3. 在左側導覽窗格的 Auto Scaling 下，選擇 Auto Scaling 群組。
4. 在頁面頂端附近選擇啟動組態。出現確認提示時，請選擇檢視啟動組態，以確認您要檢視啟動組態頁面。
5. 選擇 Create launch configuration (建立啟動組態)，然後輸入啟動組態的名稱。
6. 針對 Amazon Machine Image (AMI)，請選擇 AMI。若要尋找特定 AMI，您可以[尋找合適的 AMI](#)，記下其 ID，然後輸入 ID 作為搜尋條件。

若要獲取 Amazon Linux 2 AMI 的 ID：

- a. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
 - b. 在左側導覽窗格的執行個體下，選擇執行個體，然後選擇啟動執行個體。
 - c. 在 Quick Start 索引標籤中的 Choose an Amazon Machine Image (選擇 Amazon Machine Image) 頁面上，請注意 Amazon Linux 2 AMI (HVM) 旁的 AMI ID。
7. 針對 Instance type (執行個體類型)，為執行個體選取硬體組態。
 8. 在 Additional configuration (其他組態) 下，請注意下列欄位：
 - a. (選用) 針對 Purchasing option (購買選項)，您可以選擇 Request Spot Instances (請求 Spot 執行個體)，便能以 Spot 價格 (上限為隨需價格) 請求 Spot 執行個體。您也可以為 Spot 執行個體指定每執行個體小時的最高價格。

Note

Spot 執行個體較隨需執行個體更符合成本效益，只要您可以接受應用程式的執行時間可變動且可中斷。如需詳細資訊，請參閱[為具備容錯能力和靈活性的應用程式請求 Spot 執行個體](#)。

- b. (選用) 針對 IAM instance profile (執行個體描述檔)，請選擇與執行個體相關聯的角色。如需詳細資訊，請參閱[在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)。

- c. (選用) 針對 Monitoring (監控)，請選擇是否啟用詳細監控，以便讓執行個體以 1 分鐘的間隔將指標資料發佈到 Amazon CloudWatch。需支付額外費用。如需詳細資訊，請參閱[設定 Auto Scaling 執行個體的監控](#)。
 - d. (選用) 針對 Advanced details (進階詳細資訊)、User data (使用者資料)，您可以指定使用者資料，以便在啟動期間設定執行個體，或在執行個體啟動後執行組態指令碼。
 - e. (選用) 針對 Advanced details (進階詳細資訊)、IP address type (IP 地址類型)，請選擇是否要指派 [public IP address](#) (公有 IP 地址) 至群組的執行個體。若未設定數值，則會依預設使用啟動執行個體之子網路的自動指派公有 IP 設定。
9. (選用) 針對 Storage (volumes) (儲存 (磁碟區))，若不需要額外的儲存空間，則可略過此部分。否則，除了 AMI 所指定的磁碟區，若還要指定連接至執行個體的磁碟區，請選擇 Add new volume (新增磁碟區)。然後為 Devices (裝置)、Snapshot (快照)、Size (規模)、Volume type (磁碟區類型)、IOPS、Throughput (輸送量)、Delete on termination (終止時刪除) 以及 Encrypted (加密) 選擇所需的選項和相關數值。
10. 針對 Security groups (安全群組)，請建立或選取要與群組執行個體相關聯的安全群組。如果選擇 Create a new security group (建立新的安全群組) 選項，則會為 Amazon EC2 Linux 執行個體設定預設 SSH 規則。為 Amazon EC2 Windows 執行個體設定預設 RDP 規則。
11. 針對 Key pair (login) (金鑰對 (登入))，請選擇 Key pair options (金鑰對選項) 下的某個選項。

若您已經設定 Amazon EC2 執行個體金鑰對，則可以在這裡選擇它。

如果您還沒有 Amazon EC2 執行個體金鑰對，則請選擇 Create a new key pair (建立新的金鑰對)，然後給予可輕鬆辨識的名稱。選擇 Download Key Pair (下載金鑰對)，將金鑰對下載至電腦。

 Important

如果您需要連線到執行個體，請勿選取 Proceed without a key pair (沒有金鑰對仍繼續作業)。

12. 選取確認核取方塊，然後選擇 Create launch configuration (建立啟動組態)。

從現有的啟動組態建立啟動組態 (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在頂端導覽列上，選取您的 AWS 區域。
3. 在左側導覽窗格的 Auto Scaling 下，選擇 Auto Scaling 群組。

4. 在頁面頂端附近選擇啟動組態。出現確認提示時，請選擇檢視啟動組態，以確認您要檢視啟動組態頁面。
5. 選取啟動組態，然後選擇 Actions (動作)、Copy launch configuration (複製啟動範本)。此會以和原始相同的選項來設定新的啟動組態，但會在名稱中加入「Copy」。
6. 在 Copy Launch Configuration (複製啟動組態) 頁面，視需要編輯組態選項，然後選擇 Create launch configuration (建立啟動組態)。

使用命令列建立啟動組態

您可以使用下列其中一項命令：

- [create-launch-configuration](#) (AWS CLI)
- [New-ASLaunchConfiguration](#) (AWS Tools for Windows PowerShell)

設定執行個體中繼資料選項

Amazon EC2 Auto Scaling 支援在啟動組態中設定執行個體中繼資料服務 (IMDS)。此功能可讓您選擇使用啟動組態來設定 Auto Scaling 群組中的 Amazon EC2 執行個體，以便請求執行個體中繼資料服務第 2 版 (IMDSv2)，這是請求執行個體中繼資料的工作階段導向方法。如需 IMDSv2 優勢的詳細資訊，請參閱 AWS 部落格中關於[提升 EC2 執行個體中繼資料服務防禦深度的增強功能](#)的文章。

您可以將 IMDS 設定為支援 IMDSv2 和 IMDSv1 (預設)，或請求使用 IMDSv2。如果您使用 AWS CLI 或其中一個 SDKs 來設定 IMDS，則必須使用最新版本的 AWS CLI 或 SDK 來要求使用 IMDSv2。

您可以設定下列啟動組態項目：

- 要求執行個體中繼資料時需要 IMDSv2
- 指定 PUT 回應跳轉限制
- 關閉對執行個體中繼資料的存取

您可以在下列主題中找到有關設定執行個體中繼資料服務的詳細資訊：Amazon EC2 使用者指南中的[設定執行個體中繼資料服務](#)。

使用下列程序在啟動組態中設定 IMDS 選項。在建立啟動組態後，您可以將其與您的 Auto Scaling 群組建立關聯。若將新的啟動組態與現有 Auto Scaling 群組建立關聯，則現有啟動組態會與 Auto Scaling 群組取消關聯，而且需要取代現有執行個體，才能使用您在新啟動組態中指定的 IMDS 選項。如需詳細資訊，請參閱[變更 Auto Scaling 群組的啟動組態](#)。

在啟動組態中設定 IMDS (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在頂端導覽列上，選取您的 AWS 區域。
3. 在左側導覽窗格的 Auto Scaling 下，選擇 Auto Scaling 群組。
4. 在頁面頂端附近選擇啟動組態。出現確認提示時，請選擇檢視啟動組態，以確認您要檢視啟動組態頁面。
5. 請選擇 Create launch configuration (建立啟動組態)，再以一般方式建立啟動組態。在其中加入 Amazon Machine Image (AMI) 的 ID、執行個體類型、可選用的金鑰對、一或多個安全群組，以及任何適用於您執行個體的額外 EBS 磁碟區或執行個體存放區。
6. 若要為與此啟動組態相關聯的所有執行個體設定執行個體中繼資料選項，請在 Additional configuration (其他組態) 中的 Advanced details (進階詳細資訊) 下執行下列操作：
 - a. 針對 Metadata accessible (中繼資料可存取)，請選擇啟用或停用對執行個體中繼資料服務的 HTTP 端點的存取。根據預設，HTTP 端點為啟用狀態。若您選擇停用端點，則會關閉對執行個體中繼資料的存取。您可以指定條件，僅在啟用 HTTP 端點時才需要 IMDSv2。
 - b. 針對 Metadata version (中繼資料版本)，您可以選擇在請求執行個體中繼資料時請求使用執行個體中繼資料服務第 2 版 (IMDSv2)。若不指定數值，則預設會同時支援 IMDSv1 和 IMDSv2。
 - c. 針對 Metadata token response hop limit (中繼資料字符回應跳轉限制)，您可以為中繼資料字符設定允許的網路躍點數目。若未指定數值，則預設值為 1。
7. 完成這些設定後，請選擇 Create launch configuration (建立啟動組態)。

使用 在啟動組態中使用 IMDSv2 AWS CLI

使用 `--metadata-options` 設定為 `HttpTokens=required` 的下列 [create-launch-configuration](#) 命令。在為 `HttpTokens` 指定數值時，您也必須將 `HttpEndpoint` 設定為啟用。由於中繼資料擷取要求設定為需要安全字符標頭，因此在請求執行個體中繼資料時，即會在執行個體中請求使用 IMDSv2。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc-with-imdsv2 \  
  --image-id ami-01e24be29428c15b2 \  
  --instance-type t2.micro \  
  ...  
  --metadata-options "HttpEndpoint=enabled,HttpTokens=required"
```

關閉對執行個體中繼資料的存取

使用下列 [create-launch-configuration](#) (建立啟動組態) 命令來關閉對執行個體中繼資料的存取。您稍後可以使用 [modify-instance-metadata-options](#) 命令來重新開啟存取。

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

使用 EC2 執行個體建立啟動組態

您也可以選擇使用執行中 EC2 執行個體的屬性來建立啟動組態。

從頭建立啟動組態和從現有的 EC2 執行個體建立啟動組態是有差別的。當您從頭建立啟動組態，您會指定映像 ID、執行個體類型、選用的資源 (例如儲存裝置)，以及選用的設定 (像是監控)。在從執行中的執行個體建立啟動組態時，Amazon EC2 Auto Scaling 會從指定的執行個體獲得啟動組態的屬性。屬性也會衍生自從中啟動執行個體的 AMI 區塊型設備映射，而忽略在啟動後新增的任何其他額外區塊型儲存設備。

在使用執行中的執行個體建立啟動組態時，您可以藉由指定它們為相同請求的一部分覆寫下列屬性：AMI、區塊型儲存設備、金鑰對、執行個體描述檔、執行個體類型、核心、執行個體監控、安置租用、Ramdisk、安全群組、Spot (最高) 價格、使用者資料、執行個體是否具備公有 IP 地址，以及執行個體是否為 EBS 最佳化。

Note

若指定的執行個體具備目前不受啟動組態支援的屬性，則 Auto Scaling 群組所啟動的執行個體可能不同於原始 EC2 執行個體。

Important

用於啟動指定的執行個體的 AMI 必須仍然存在。

主題

- [從 EC2 執行個體建立啟動組態 \(AWS CLI\)](#)
- [從執行個體建立啟動組態，並覆寫區塊型儲存設備 \(AWS CLI\)](#)

- [建立啟動組態並覆寫執行個體類型 \(AWS CLI\)](#)

從 EC2 執行個體建立啟動組態 (AWS CLI)

使用下列 [create-launch-configuration](#) 命令，利用與該執行個體相同的屬性從執行個體建立啟動組態。任何在啟動後新增的區塊型儲存設備都會被忽略。

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

您可以使用下列 [describe-launch-configurations](#) 命令來描述啟動組態，並確認其屬性與執行個體的屬性相符：

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

以下是回應範例。

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


}

從執行個體建立啟動組態，並覆寫區塊型儲存設備 (AWS CLI)

在預設情況下，Amazon EC2 Auto Scaling 會使用您指定的 EC2 執行個體的屬性來建立啟動組態。不過，區塊型儲存設備來自用於啟動執行個體的 AMI，而非執行個體。若要新增區塊型儲存設備到啟動組態，請覆寫啟動組態的區塊型設備映射。

使用下列 [create-launch-configuration](#) 命令來利用 EC2 執行個體建立啟動組態，但搭配自訂區塊型設備映射。

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

使用下列 [describe-launch-configurations](#) 命令來描述啟動組態，並確認其使用您的自訂區塊型設備映射。

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

下列範例回應描述啟動組態。

```
{
  "LaunchConfigurations": [
    {
      "UserData": null,
      "EbsOptimized": false,
      "LaunchConfigurationARN": "arn",
      "InstanceMonitoring": {
        "Enabled": false
      },
      "ImageId": "ami-c49c0dac",
      "CreatedTime": "2015-01-07T14:51:26.065Z",
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": {
            "SnapshotId": "snap-3decf207"
          }
        }
      ]
    }
  ]
}
```



```

        }
    },
    {
        "DeviceName": "/dev/sdf",
        "Ebs": {
            "SnapshotId": "snap-eed6ac86"
        }
    }
],
"KeyName": "my-key-pair",
"SecurityGroups": [
    "sg-8637d3e3"
],
"LaunchConfigurationName": "my-lc-from-instance-bdm",
"KernelId": null,
"RamdiskId": null,
"InstanceType": "t1.micro",
"AssociatePublicIpAddress": true
}
]
}

```

建立啟動組態並覆寫執行個體類型 (AWS CLI)

在預設情況下，Amazon EC2 Auto Scaling 會使用您指定的 EC2 執行個體的屬性來建立啟動組態。根據您的要求，您可能想要覆寫執行個體的屬性，並使用您需要的值。例如，您可以覆寫執行個體類型。

使用下列 [create-launch-configuration](#) 命令來利用 EC2 執行個體建立啟動組態，但執行個體類型 (例如：t2.medium) 與該執行個體不同 (例如：t2.micro)。

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

使用下列 [describe-launch-configurations](#) 命令來描述啟動組態，並確認已覆寫執行個體類型：

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

下列範例回應描述啟動組態。

```
{
```

```
"LaunchConfigurations": [
  {
    "UserData": null,
    "EbsOptimized": false,
    "LaunchConfigurationARN": "arn",
    "InstanceMonitoring": {
      "Enabled": false
    },
    "ImageId": "ami-05355a6c",
    "CreatedTime": "2014-12-29T16:14:50.382Z",
    "BlockDeviceMappings": [],
    "KeyName": "my-key-pair",
    "SecurityGroups": [
      "sg-8422d1eb"
    ],
    "LaunchConfigurationName": "my-lc-from-instance-changetype",
    "KernelId": "null",
    "RamdiskId": null,
    "InstanceType": "t2.medium",
    "AssociatePublicIpAddress": true
  }
]
```

變更 Auto Scaling 群組的啟動組態

Important

對於那些尚未從啟動組態遷移至啟動範本的客戶，我們會提供啟動組態的資訊。如需將 Auto Scaling 群組遷移至啟動範本的詳細資訊，請參閱 [遷移 Auto Scaling 群組以啟動範本](#)。

本主題說明如何將不同的啟動組態與 Auto Scaling 群組建立關聯。

變更啟動組態後，任何新的執行個體都會使用新的組態選項啟動，但現有的執行個體不會受到影響。如需詳細資訊，請參閱[更新 Auto Scaling 執行個體](#)。

變更 Auto Scaling 群組的啟動組態 (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在左側導覽窗格的 Auto Scaling 下，選擇 Auto Scaling 群組。

3. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在 Details (詳細資訊) 索引標籤上，請依序選擇 Launch configuration (啟動組態) 和 Edit (編輯)。

5. 針對啟動組態，選擇啟動組態。

6. 完成後，請選擇 Update (更新)。

使用命令列變更 Auto Scaling 群組的啟動組態

您可以使用下列其中一項命令：

- [update-auto-scaling-group](#) (AWS CLI)
- [Update-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

Auto Scaling 群組

Note

如果您是初次使用 Auto Scaling 群組，請執行[建立您的第一個 Auto Scaling 群組](#)教學課程中的步驟以開始使用，並查看 Auto Scaling 群組在群組中的執行個體終止時如何回應。

Auto Scaling 群組包含一群 EC2 執行個體，該群就自動擴展和管理目的而言被視為一個邏輯分組。Auto Scaling 群組也讓您可以使用 Amazon EC2 Auto Scaling 功能，例如運作狀態檢查替換和擴展政策。維持 Auto Scaling 群組中的執行個體數量和自動擴展都是 Amazon EC2 Auto Scaling 服務的核心功能。

Auto Scaling 群組的大小取決於您設為所需容量的執行個體數目。您可以手動或使用自動擴展調整其大小，以滿足需求。

Auto Scaling 群組開始作業時會啟動足夠的執行個體，以滿足自己所需的容量。它會定期對群組中的執行個體執行運作狀態檢查，以保持執行個體的數量。即使有一個執行個體的運作狀態不佳，Auto Scaling 群組仍會繼續維持固定數量的執行個體。如果執行個體運作狀態變成不佳，該群組會終止運作狀態不佳的執行個體，並啟動另一個執行個體加以取代。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

您可以使用擴展政策，動態地增加或減少群組中執行個體的數量，以滿足不斷變化的條件。當擴展政策生效時，Auto Scaling 群組會在您指定的最小和最大容量值之間調整群組的所需容量，並視需要啟動或終止執行個體。您也可以依照時間表進行擴展。如需詳細資訊，請參閱[選擇您的擴展方法](#)。

建立 Auto Scaling 群組時，可以選擇啟動隨需執行個體、Spot 執行個體或這兩者。只有在使用時，您才能為 Auto Scaling 群組指定多個購買選項launch template。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

Spot 執行個體可讓您以比隨需價格優惠許多的折扣存取未使用的 EC2 容量。如需詳細資訊，請參閱[Amazon EC2 Spot 執行個體](#)。Spot 執行個體和隨需執行個體之間的主要差異：

- Spot 執行個體的價格根據需求而有所不同
- 隨著 Spot 執行個體的可用性或價格變化，Amazon EC2 可以終止個別 Spot 執行個體

當 Spot 執行個體終止時，Auto Scaling 群組會嘗試啟動替代的執行個體來維持群組的所需容量。

執行個體啟動時，如果您指定了多個可用區域，所需容量會分配到這些可用區域。如果發生了某個擴展動作，Amazon EC2 Auto Scaling 會自動維護您指定的所有可用區域之間的平衡。

目錄

- [使用啟動範本建立 Auto Scaling 群組](#)
- [使用啟動組態建立 Auto Scaling 群組](#)
- [同步啟動執行個體](#)
- [更新 Auto Scaling 群組](#)
- [標記 Auto Scaling 群組和執行個體](#)
- [執行個體維護政策](#)
- [Amazon EC2 Auto Scaling lifecycle hook](#)
- [使用暖集區縮短開機時間的應用程式延遲](#)
- [Auto Scaling 群組區域轉移](#)
- [Auto Scaling 群組可用區域分佈](#)
- [從 Auto Scaling 群組分離或連接執行個體](#)
- [從 Auto Scaling 群組暫時移除執行個體](#)
- [刪除 Auto Scaling 基礎設施](#)

使用啟動範本建立 Auto Scaling 群組

如果您已建立啟動範本，則可以建立將啟動範本用作其 EC2 執行個體的組態範本的 Auto Scaling 群組。啟動範本指定了執行個體的 AMI ID、執行個體類型、金鑰對、安全群組和區塊型裝置映射等資訊。如需建立啟動範本的相關資訊，請參閱[為 Auto Scaling 群組建立啟動範本](#)。

您必須擁足夠的權限來建立 Auto Scaling 群組。您還必須擁有足夠的權限來建立 Amazon EC2 Auto Scaling 用於代表您執行動作的服務連結角色 (如果該角色尚不存在)。有關管理員授予您權限時可用作參考的 IAM 政策範例，請參閱 [身分型政策範例](#) 和 [在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量](#)。

目錄

- [使用啟動範本建立 Auto Scaling 群組](#)
- [使用 Amazon EC2 啟動精靈建立 Auto Scaling 群組](#)
- [具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)

使用啟動範本建立 Auto Scaling 群組

建立 Auto Scaling 群組時，您必須指定必要資訊，以設定 Amazon EC2 執行個體、執行個體的可用區域和 VPC 子網路、所需的容量以及最小和最大容量限制。

若要設定 Auto Scaling 群組啟動的 Amazon EC2 執行個體，您可以指定啟動範本或啟動組態。下列程序示範如何使用啟動範本建立 Auto Scaling 群組。

先決條件

- 您必須已經建立啟動範本。如需詳細資訊，請參閱[為 Auto Scaling 群組建立啟動範本](#)。

使用啟動範本建立 Auto Scaling 群組 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在畫面頂端的導覽列上，選擇 AWS 區域 您在建立啟動範本時所使用的相同。
3. 選擇建立 Auto Scaling 群組。
4. 在 Choose launch template or configuration (選擇啟動範本或組態) 頁面上，執行下列動作：
 - a. 針對 Auto Scaling 群組名稱，輸入 Auto Scaling 群組的名稱。
 - b. 針對啟動範本，選擇現有的啟動範本。
 - c. 針對 Launch template version (啟動範本版本)，選擇 Auto Scaling 群組擴增時使用啟動範本的預設、最新或特定版本。
 - d. 確認您的啟動範本支援您打算使用的所有選項，然後選擇 Next (下一步)。
5. 在選擇執行個體啟動選項頁面上，如果您未使用多個執行個體類型，可以略過執行個體類型需求區段，使用在啟動範本中指定的 EC2 執行個體類型。

若要使用多種執行個體類型，請參閱 [具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

6. 在 Network (網路) 下，對於 VPC，選擇 VPC。Auto Scaling 群組必須與您在啟動範本中指定的安全群組建立在同一個 VPC 中。
7. 對於 (子網路)、Availability Zones and subnets (可用區域和子網路)，請選擇指定 VPC 中的一個或多個子網路。在多個可用區域中，請使用子網路，以獲得高可用性。如需詳細資訊，請參閱[選擇 VPC 子網路時的注意事項](#)。
8. 針對可用區域分佈，選取分佈策略。如需詳細資訊，請參閱[Auto Scaling 群組可用區域分佈](#)。

9. 如果您建立了具有指定執行個體類型的啟動範本，則可以繼續執行下一步，以建立使用啟動範本中執行個體類型的 Auto Scaling 群組。

或者，如果在啟動範本中未指定任何執行個體類型，或者您想要使用多個執行個體類型進行自動擴展，則可以選擇 Override launch template (覆寫啟動範本) 選項。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

10. 選擇 Next (下一步)，並繼續進行下一個步驟。

或者，您可以接受其他的預設值，然後選擇 Skip to review (跳過以進行檢閱)。

11. (選用) 在與其他服務整合頁面上，設定下列選項，然後選擇下一步：

- a. 針對負載平衡，選擇是否要將 Auto Scaling 群組連接至負載平衡器。如需詳細資訊，請參閱[Elastic Load Balancing](#)。
- b. 針對 VPC Lattice 整合選項，選擇是否使用 VPC Lattice。如需詳細資訊，請參閱[使用 VPC Lattice 目標群組管理流量流程](#)。
- c. 針對 Amazon Application Recovery Controller (ARC) 區域轉移，選取核取方塊以啟用區域轉移。如需詳細資訊，請參閱[Auto Scaling 群組區域轉移](#)。
 - 如果您啟用區域轉移，請針對運作狀態檢查行為，選取忽略運作狀態不佳或取代運作狀態不佳。如需詳細資訊，請參閱[Auto Scaling 群組的區域轉移運作方式](#)。
- d. 在運作狀態檢查下，針對其他運作狀態檢查類型，選取開啟 Amazon EBS 運作狀態檢查。如需詳細資訊，請參閱[使用運作狀態檢查監控 Amazon EBS 磁碟區受損的 Auto Scaling 執行個體](#)。
- e. 對於運作狀態檢查寬限期，輸入時間量，以秒為單位。此時間量是 Amazon EC2 Auto Scaling 在進入 InService 狀態後檢查執行個體的運作狀態所需的等待時間。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。

12. (選用) 在設定群組大小和擴展頁面上，設定下列選項，然後選擇下一步：

- a. 針對所需容量，在群組大小下輸入要啟動之執行個體的初始數量。
- b. 在擴展、擴展限制下，如果您的所需容量新值大於所需容量下限和所需容量上限，則所需容量上限會自動增加至新的所需容量值。您可視需要變更這些限制。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
- c. 針對 Automatic scaling，選擇是否要建立目標追蹤擴展政策。您也可以在建立 Auto Scaling 群組後建立此政策。

如果您選擇目標追蹤擴展政策，請依照 [建立目標追蹤擴展政策](#) 中的指示建立政策。

- d. 在執行個體維護政策下，選擇您是否要建立執行個體維護政策。您也可以在建立 Auto Scaling 群組後建立此政策。請遵循 [設定執行個體維護政策](#) 中的指示建立政策。
 - e. 在其他容量設定下，容量保留偏好設定下，選擇您是否要使用容量保留偏好設定。如需詳細資訊，請參閱[使用容量預留在特定可用區域中預留容量](#)。
 - f. 在其他設定下，執行個體縮減保護，選擇是否啟用執行個體縮減保護。如需詳細資訊，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。
 - g. 針對監控，選擇是否啟用 CloudWatch 群組指標集合。這些指標提供了可作為潛在問題之指標的度量，例如終止執行個體數量或擱置執行個體的數量。如需詳細資訊，請參閱[監控 Auto Scaling 群組和執行個體的 CloudWatch 指標](#)。
 - h. 針對預設執行個體暖機期，選取此選項，然後選擇應用程式的暖機期。如果您正在建立具有擴展政策的 Auto Scaling 群組，則預設執行個體暖機功能會改進用於動態擴展的 Amazon CloudWatch 指標。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。
13. (選用) 在新增通知頁面上，設定通知，然後選擇下一步。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 的 Amazon SNS 通知選項 Amazon EC2 Auto Scaling](#)。
 14. (選用) 在新增標籤頁面上，選擇新增標籤，為每個標籤提供標籤索引鍵和值，然後選擇下一步。如需詳細資訊，請參閱[標記 Auto Scaling 群組和執行個體](#)。
 15. 在 Review (檢閱) 頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

使用命令列建立 Auto Scaling 群組

您可以使用下列其中一項命令：

- [create-auto-scaling-group](#) (AWS CLI)
- [New-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

使用 Amazon EC2 啟動精靈建立 Auto Scaling 群組

以下程序顯示如何透過使用 Amazon EC2 主控台內的 Launch instance (啟動執行個體) 精靈建立 Auto Scaling 群組。此選項會自動使用 Launch instance (啟動執行個體) 精靈的特定組態詳細資訊，填入啟動範本。

Note

精靈不會使用您指定的執行個體數量填入 Auto Scaling 群組；它只會使用 Amazon Machine Image (AMI) ID 和執行個體類型填入啟動範本。使用 Create Auto Scaling group (建立 Auto Scaling 群組) 精靈指定要啟動的執行個體數量。

AMI 提供設定執行個體的必要資訊。當您需要多個具有相同組態的執行個體時，可以從單一 AMI 啟動多個執行個體。建議使用已安裝應用程式的自訂 AMI，以避免在重新啟動屬於 Auto Scaling 群組的執行個體時終止執行個體。要將自訂 AMI 與 Amazon EC2 Auto Scaling 搭配使用，您必須首先從自訂執行個體建立 AMI，然後使用 AMI 為您的 Auto Scaling 群組建立啟動範本。

先決條件

- 您必須在計劃建立 Auto Scaling 群組的相同 AWS 區域中建立自訂 AMI。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[建立 AMI](#)。

使用自訂 AMI 作為範本

在本節中，您可以使用 Amazon EC2 Launch Wizard，透過您的自訂 AMI 自動填入啟動範本。或者，若要從頭開始設定啟動範本，或是您可為啟動範本設定的參數詳細描述，請參閱[建立啟動範本 \(主控台\)](#)。

使用自訂 AMI 作為範本

- 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
- 在畫面頂端的導覽列上，AWS 區域 會顯示目前的 。選取要在其中啟動 Auto Scaling 群組的區域。
- 在導覽窗格中，選擇 Instances (執行個體)。
- 選擇 Launch instance (啟動執行個體)，並執行下列動作：
 - 在 Name and tags (名稱和標籤) 下，將 Name (名稱) 保留空白。名稱不是用於建立啟動範本的資料的一部分。
 - 在 Application and OS Images (Amazon Machine Image) (應用程式和作業系統映像 (Amazon Machine Image)) 下，選擇 Browse more AMIs (瀏覽更多 AMI) 以瀏覽完整的 AMI 目錄。
 - 選擇 My AMIs (我的 AMI)，找出您之前建立的 AMI，然後選擇 Select (選取)。
 - 在 Instance type (執行個體類型)，選擇執行個體類型。

 Note

選擇建立 AMI 時使用的相同執行個體類型或更強大的執行個體類型。

- e. 在畫面右側的 Summary (摘要) 下，針對 Number of instances (執行個體數量)，輸入任意數量。您在此輸入的數量並不重要。在建立 Auto Scaling 群組時，您將指定您要啟動的執行個體數量。

在 Number of instances (執行個體數量) 欄位下，將顯示訊息：When launching more than 1 instance, consider EC2 Auto Scaling (當啟動多個執行個體時，請考慮 EC2 Auto Scaling)。

- f. 選擇 consider EC2 Auto Scaling (考慮 EC2 Auto Scaling) 超連結文字。
- g. 在 Launch into Auto Scaling Group (啟動 Auto Scaling 群組) 確認對話方塊中，選擇 Continue (繼續) 前往 Create launch template (建立啟動範本) 頁面，頁面中已填入您在啟動執行個體精靈中選取的 AMI 和執行個體類型。

選擇 Continue (繼續) 後，Create launch template (建立啟動範本) 頁面隨即開啟。請依照以下程序來完成啟動範本的建立。

建立啟動範本

1. 在 Launch template name and description (啟動範本名稱和描述) 下，輸入新啟動範本的名稱和描述。
2. (選用) 在 Key pair (login) (金鑰對 (登入)) 下，對於 Key pair name (金鑰對名稱)，選擇要在連線執行個體時使用之先前建立的金鑰對名稱，例如使用 SSH。
3. (選用) 在 Network settings (網路設定) 下，對於 Security groups (安全群組)，選擇一或多個之前建立的[安全群組](#)。
4. (選用) 在 Configure storage (設定儲存) 下，更新儲存組態。預設儲存組態會由 AMI 和執行個體類型判定。
5. 完成啟動範本的設定之後，請選擇 Create launch template (建立啟動範本)。
6. 在確認頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

建立 Auto Scaling 群組

Note

本主題的其餘部分描述了建立 Auto Scaling 群組的基本程序。有關可為 Auto Scaling 群組設定參數的詳細描述，請參閱 [使用啟動範本建立 Auto Scaling 群組](#)。

選擇 Create Auto Scaling group (建立 Auto Scaling 群組) 後，Create Auto Scaling group (建立 Auto Scaling 群組) 精靈會隨即開啟。請執行以下程序來建立 Auto Scaling 群組。

建立 Auto Scaling 群組

1. 在 Choose launch template or configuration (選擇啟動範本或組態) 頁面上，輸入 Auto Scaling 群組的名稱。
2. 已為您選取了您建立的啟動範本。

針對啟動範本版本，選擇 Auto Scaling 群組擴增時使用啟動範本的預設、最新或特定版本。

3. 選擇 Next (下一步)，並繼續進行下一個步驟。
4. 在選擇執行個體啟動選項頁面上，如果您未使用多個執行個體類型，可以略過執行個體類型需求區段，使用在啟動範本中指定的 EC2 執行個體類型。

若要使用多種執行個體類型，請參閱 [具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

5. 在 Network (網路) 下，對於 VPC，選擇 VPC。Auto Scaling 群組必須與您在啟動範本中指定的安全群組建立在同一個 VPC 中。

Tip

如果您沒有在啟動範本中指定安全群組，則您的執行個體將使用您指定之 VPC 中的預設安全群組啟動。根據預設，此安全群組不允許來自外部網路的傳入流量。

6. 對於 (子網路)、Availability Zones and subnets (可用區域和子網路)，請選擇指定 VPC 中的一個或多個子網路。
7. 針對可用區域分佈，選取分佈策略。如需詳細資訊，請參閱 [Auto Scaling 群組可用區域分佈](#)。
8. 選擇 Next (下一頁) 兩次以前往 Configure group size and scaling policies (設定群組大小和擴展政策) 頁面。
9. 在群組大小下，定義所需容量 (建立 Auto Scaling 群組後立即啟動的初始執行個體數量)。

10. 在擴展區段的擴展限制下，如果所需容量的新值大於所需容量下限和所需容量上限，則所需容量上限會自動增加到新的所需容量值。您可視需要變更這些限制。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
11. 選擇 Skip to review (跳過以進行檢閱)。
12. 在 Review (檢閱) 頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

後續步驟

您可以透過檢視活動歷史記錄來查看 Auto Scaling 群組是否已正確建立。在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 下方，Status (狀態) 欄會顯示 Auto Scaling 群組是否已成功啟動執行個體。如果執行個體無法啟動或啟動後立即終止，則請參閱以下主題以了解可能的原因和解決方案：

- [Amazon EC2 Auto Scaling 故障診斷：EC2 執行個體啟動失敗](#)
- [Amazon EC2 Auto Scaling 故障診斷：AMI 問題](#)
- [對 Amazon EC2 Auto Scaling 中運作狀態不佳的執行個體進行故障診斷](#)

如果需要，您現在可以連接與 Auto Scaling 群組相同區域中的負載平衡器。如需詳細資訊，請參閱[使用 Elastic Load Balancing 在 Auto Scaling 群組中分配傳入應用程式流量](#)。

具有多種執行個體類型和購買選項的 Auto Scaling 群組

您可以在單一 Auto Scaling 群組內啟動和自動擴展隨需執行個體和 Spot 執行個體組成的機群。除了獲得使用 Spot 執行個體的折扣之外，您還可以使用預留執行個體或 Savings Plans 來獲得一般隨需執行個體定價的折扣。這些因素可協助您將 EC2 執行個體的費用節省最佳化，並獲得應用程式所需的規模與效能。

與 EC2 隨需價格相比，Spot 執行個體是以極低折扣提供的備用容量。如能彈性決定應用程式執行的時機以及應用程式是否可以中斷，則競價型執行個體會是一項具成本效益的選擇。它們可用於各種容錯和靈活的應用程式。範例包括無狀態 Web 伺服器、API 端點、大數據和分析應用程式、容器化工作負載、CI/CD 管道、高效能和高輸送量運算 (HPC/HTC)、轉譯工作負載和其他彈性工作負載。

如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[執行個體購買選項](#)。

主題

- [建立混合執行個體群組的設定概觀](#)
- [多種執行個體類型的配置策略](#)

- [使用屬性型執行個體類型選取項目建立混合執行個體群組](#)
- [手動選擇執行個體類型以建立混合執行個體群組](#)
- [設定 Auto Scaling 群組以使用執行個體權重](#)
- [使用多個啟動範本](#)

建立混合執行個體群組的設定概觀

本主題提供建立 Auto Scaling 混合執行個體群組的概觀和最佳實務。

目錄

- [概要](#)
- [執行個體類型彈性](#)
- [可用區域彈性](#)
- [Spot 最高價格](#)
- [主動容量重新平衡](#)
- [擴展行為](#)
- [執行個體類型的區域可用性](#)
- [相關資源](#)
- [限制](#)

概要

若要建立混合執行個體群組，您有兩種選擇：

- [屬性型執行個體類型選擇](#) – 定義您的運算需求，以根據執行個體的特定屬性自動選擇執行個體類型。
- [手動執行個體類型選擇](#) – 手動選擇適合您工作負載的執行個體類型。

Manual selection

下列步驟說明如何透過手動選擇執行個體類型來建立混合執行個體群組：

1. 選擇包含啟動 EC2 執行個體的參數的啟動範本。啟動範本中的參數是選用的，但如果啟動範本中缺少 `amiId`；(AMI) ID，則 Amazon EC2 Auto Scaling 無法啟動執行個體。

2. 選擇覆寫啟動範本的選項。
3. 手動選擇適合您工作負載的執行個體類型。
4. 指定要啟動的隨需執行個體和 Spot 執行個體百分比。
5. 選擇決定 Amazon EC2 Auto Scaling 如何利用可能的執行個體類型滿足隨需和 Spot 容量所需容量的分配策略。
6. 選擇您要在其中啟動執行個體的可用區域和 VPC 子網路。
7. 指定群組的初始大小 (所需容量) 以及群組的最小和最大大小。

有必要進行覆寫動作，以覆寫啟動範本中宣告的執行個體類型，並使用內嵌在 Auto Scaling 群組本身資源定義中的多個執行個體類型。如需可用執行個體類型的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[執行個體類型](#)。

您也可以為每個執行個體類型設定下列選用參數：

- `LaunchTemplateSpecification` – 您可以視需要將不同的啟動範本指派給執行個體類型。此選項目前在主控台中無法使用。如需詳細資訊，請參閱[使用多個啟動範本](#)。
- `WeightedCapacity` – 您決定執行個體相對於群組中其他執行個體計入所需容量的程度。如果為某個執行個體類型指定了 `WeightedCapacity` 值，則必須為所有執行個體類型指定 `WeightedCapacity` 值。預設情況下，每個執行個體會計入所需容量中計為 1。如需詳細資訊，請參閱[設定 Auto Scaling 群組以使用執行個體權重](#)。

Attribute-based selection

若要讓 Amazon EC2 Auto Scaling 根據執行個體的特定屬性自動選擇執行個體類型，請使用以下步驟指定運算要求來建立混合執行個體群組：

1. 選擇包含啟動 EC2 執行個體的參數的啟動範本。啟動範本中的參數是選用的，但如果啟動範本中缺少 `amiId`；(AMI) ID，則 Amazon EC2 Auto Scaling 無法啟動執行個體。
2. 選擇覆寫啟動範本的選項。
3. 指定與您的運算要求 (例如 vCPU 和記憶體要求) 相符的執行個體屬性。
4. 指定要啟動的隨需執行個體和 Spot 執行個體百分比。
5. 選擇決定 Amazon EC2 Auto Scaling 如何利用可能的執行個體類型滿足隨需和 Spot 容量所需容量的分配策略。
6. 選擇您要在其中啟動執行個體的可用區域和 VPC 子網路。
7. 指定群組的初始大小 (所需容量) 以及群組的最小和最大大小。

有必要進行覆寫動作，以覆寫啟動範本中宣告的執行個體類型，並使用一組描述您運算要求的執行個體屬性。對於支援屬性，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [InstanceRequirements](#)。或者，您也可以使用已具有執行個體屬性定義的啟動範本。

您也可以覆寫結構中設定 `LaunchTemplateSpecification` 參數，視需要將不同的啟動範本指派給一組執行個體需求。此選項目前在主控台中無法使用。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [LaunchTemplateOverrides](#)。

根據預設，您可以將執行個體數量設定為 Auto Scaling 群組的所需容量。

或者，您可以將所需容量的值設定為 vCPU 的數量或記憶體量。若要這麼做，請使用 `CreateAutoScalingGroup` API 作業中的 `DesiredCapacityType` 屬性或 AWS 管理主控台所需的容量類型下拉式欄位。這是一個可以替代[執行個體加權](#)的有用方案。

執行個體類型彈性

若要加強可用性，請跨多個執行個體類型部署應用程式。最佳的方式是使用多個執行個體類型來滿足容量需求。這樣，如果所選可用區域中的執行個體容量不足，則 Amazon EC2 Auto Scaling 可以啟動其他執行個體類型。

如果 Spot 執行個體的執行個體容量不足，則 Amazon EC2 Auto Scaling 會繼續嘗試從其他 Spot 執行個體集區啟動。(其使用的集區取決於您選擇的執行個體類型和分配策略。) 執行 Spot 執行個體，而非執行隨需執行個體。Amazon EC2 Auto Scaling 從而幫助您利用 Spot 執行個體節省成本。

建議您有彈性地對每個工作負載至少使用 10 個執行個體類型。選擇執行個體類型時，不要侷限於最熱門的新執行個體類型。選擇舊版執行個體類型往往可以達到減少 Spot 中斷的效果，因為隨需客戶的需求較少。

可用區域彈性

強烈建議您讓您的 Auto Scaling 群組跨越多個可用區域。使用多個可用區域，您可以設計在區域之間自動容錯移轉的應用程式，以獲得更高的彈性。

相較於單一可用區域中的群組，您可以存取更深層的 Amazon EC2 容量集區，這是一項額外的好處。因為每個可用區域內每一個執行個體類型的容量會獨立變動，因此您通常可以透過執行個體類型和可用區域的靈活性來取得更多運算容量。

如需使用多個可用區域的詳細資訊，請參閱 [範例：跨可用區域分配執行個體](#)。

Spot 最高價格

當您使用 AWS CLI 或 SDK 建立 Auto Scaling 群組時，您可以指定 SpotMaxPrice 參數。SpotMaxPrice 參數決定您願意為 Spot 執行個體支付的每小時最高價。

當您在覆寫 (或群組層級的 "DesiredCapacityType": "vcpu" 或 "DesiredCapacityType": "memory-mib") 中指定 WeightedCapacity 參數時，最高價代表單價上限，而不是整個執行個體的最高價。

強烈建議您不要指定最高價。如果您未收到 Spot 執行個體 (例如當您的最高價太低時)，則您的應用程式可能不會執行。如果您未指定最高價格，則預設的最高價格是隨需價格。您只需為您啟動的 Spot 執行個體支付 Spot 價格。您仍然可以獲得 Spot 執行個體提供的大幅折扣。因為使用 [Spot 定價模式](#) 可提供穩定的 Spot 定價，所以可以享有這些折扣。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [定價和節省](#)。

主動容量重新平衡

如果您的使用案例允許，建議您重新平衡容量。容量重新平衡會主動取代具有中斷風險的 Spot 執行個體，協助您維持工作負載可用性。

啟用容量重新平衡時，Amazon EC2 Auto Scaling 會嘗試主動取代已收到 EC2 執行個體重新平衡建議的 Spot 執行個體。這可讓您將工作負載重新平衡到沒有較高中斷風險的新 Spot 執行個體。

如需詳細資訊，請參閱 [Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體](#)。

擴展行為

建立混合執行個體群組時，預設會使用隨需執行個體。若要使用 Spot 執行個體，您必須修改群組以隨需執行個體形式啟動的百分比。您可以為隨需百分比指定從 0 到 100 的任意數量。

或者，您也可指定隨需執行個體基本數量來開始。如果這樣做，Amazon EC2 Auto Scaling 會在群組橫向擴展時啟動隨需執行個體的基本容量後啟動 Spot 執行個體。超過基本容量的項目會使用隨需百分比來決定要啟動多少隨需執行個體和 Spot 執行個體。

Amazon EC2 Auto Scaling 會將百分比轉換為同等數量的執行個體。如果產生的結果是小數，則會四捨五入到下一個整數，以便隨需執行個體使用。

下表說明了 Auto Scaling 群組大小增加和降低時的行為。

範例：擴展行為

購買選項	購買選項中的群組大小和執行中的執行個體總數			
	10	20	30	40
範例 1：以 10、50/50% 的 隨需/Spot 為基準				
On-Demand Instances (base amount)	10	10	10	10
On-Demand Instances	0	5	10	15
Spot Instances	0	5	10	15
範例 2：以 0、0/100% 的隨 需/Spot 為基準				
On-Demand Instances (base amount)	0	0	0	0
On-Demand Instances	0	0	0	0
Spot Instances	10	20	30	40
範例 3：以 0、60/40% 隨需/ Spot 為基準				
On-Demand Instances (base amount)	0	0	0	0

購買選項	購買選項中的群組大小和執行中的執行個體總數			
On-Demand Instances	6	12	18	24
Spot Instances	4	8	12	16
範例 4：以 0、100/0% 隨需/Spot 為基準				
On-Demand Instances (base amount)	0	0	0	0
On-Demand Instances	10	20	30	40
Spot Instances	0	0	0	0
範例 5：以 12、0/100% 隨需/Spot 為基準				
On-Demand Instances (base amount)	10	12	12	12
On-Demand Instances	0	0	0	0
Spot Instances	0	8	18	28

當群組大小增加時，Amazon EC2 Auto Scaling 會嘗試在指定的可用區域之間平均地平衡容量。然後，其會根據指定的分配策略啟動執行個體類型。

當群組的大小降低時，Amazon EC2 Auto Scaling 會先確定應終止兩種類型中的哪一種 (Spot 或隨需)。接著，它會嘗試在指定的可用區域之間以平衡的方式終止執行個體。這也有利於以更接近配置策

略的方式終止執行個體。若要取得有關終止政策的資訊，請參閱 [設定 Amazon EC2 Auto Scaling 的終止政策](#)。

執行個體類型的區域可用性

EC2 執行個體類型的可用性取決於您的 AWS 區域。例如，最新一代的執行個體類型也許尚無法用於特定的區域。由於不同區域的執行個體可用性差異，如果您的區域中無法使用覆寫中的多個執行個體類型，則在提出程式設計請求時可能會遇到問題。使用您區域中不可用的多個執行個體類型可能會導致請求完全失敗。若要解決此問題，請重試不同的執行個體類型的請求，確保每個執行個體類型都在區域中可用。若要依位置搜尋提供的執行個體類型，請使用 [describe-instance-type-offerings](#) 命令。如需詳細資訊，請參閱《[Amazon EC2 使用者指南](#)》中的[尋找 Amazon EC2 執行個體類型](#)。Amazon EC2

相關資源

如需 Spot 執行個體的更多最佳實務，請參閱《Amazon [EC2 使用者指南](#)》中的[EC2 Spot 的最佳實務](#)。Amazon EC2

限制

使用[混合執行個體政策](#)將覆寫新增至 Auto Scaling 群組之後，您可以使用 UpdateAutoScalingGroup API 呼叫更新覆寫，但不能將其刪除。若要完全移除覆寫，您必須先切換 Auto Scaling 群組，以使用啟動範本或啟動組態，而非混合執行個體政策。然後，您可以再次新增混合執行個體政策，而無需任何覆寫。

多種執行個體類型的配置策略

當您使用多個執行個體類型時，您可以管理 Amazon EC2 Auto Scaling 如何從可能的執行個體類型中滿足隨需和 Spot 容量。若要這麼做，您可以指定配置策略。

若要檢閱混合執行個體群組的最佳實務，請參閱 [建立混合執行個體群組的設定概觀](#)。

目錄

- [Spot 執行個體](#)
- [隨需執行個體](#)
- [配置策略如何使用權重](#)

Spot 執行個體

Amazon EC2 Auto Scaling 提供下列 Spot 執行個體的配置策略：

price-capacity-optimized (建議)

價格和容量最佳化分配策略會考慮價格和容量，來選擇最不可能中斷且價格最低的 Spot 執行個體集區。

建議您在剛開始時使用此策略。如需詳細資訊，請參閱 AWS 部落格中的[介紹 EC2 Spot price-capacity-optimized 配置策略](#)。

capacity-optimized

Amazon EC2 Auto Scaling 使用啟動中執行個體數目的最佳容量來請求來自集區的 Spot 執行個體。

使用 Spot 執行個體，定價會依據長期的供需趨勢隨時間慢慢改變。但是，容量會即時波動。capacity-optimized 策略會查看即時容量資料並預測哪些資料為最多可用，從而自動將 Spot 執行個體啟動到最可用的集區。這有助於最大程度減少工作負載的可能中斷，因為重新啟動工作和檢查點可能會導致較高的中斷成本。若要讓某些執行個體類型先啟動的機會較高，請使用 capacity-optimized-prioritized。

capacity-optimized-prioritized

您可以按照從最高到最低優先順序，對啟動範本覆寫設定執行個體類型的順序 (從清單中的第一個到最後一個)。Amazon EC2 Auto Scaling 會盡力遵守執行個體類型的優先順序，但會先針對容量進行最佳化。對於必須將中斷可能性降至最低的工作負載來說，這是一個很好的選擇，對於某些執行個體類型的偏好也很重要。如果隨需分配策略設定為 prioritized，則在滿足隨需容量時會套用相同的優先順序。

lowest-price (不建議使用)

Warning

我們不建議採用 lowest-price 策略，因為它對您的 Spot 執行個體具有最高的中斷風險。

Amazon EC2 Auto Scaling 使用可用區域內價格最低的集區來請求 Spot 執行個體，跨越您為最低價集區設定指定的 N 個 Spot 集區。例如，如果您指定四個執行個體類型和四個可用區域，則您的 Auto Scaling 群組最多可以存取 16 個 Spot 集區。(每個可用區域四個。) 如果您為分配策略指定兩個 Spot 集區 (N = 2)，則 Auto Scaling 群組 可以利用每個可用區域中最便宜的兩個集區，來滿足 Spot 容量。

只有在使用時，才能使用 lowest-price 策略 AWS CLI。

Amazon EC2 Auto Scaling 會盡最大努力從您指定的 N 個集區中提取 Spot 執行個體。但是，如果集區在滿足您的目標容量之前用完 Spot 容量，Amazon EC2 Auto Scaling 會透過從下一個最便宜的集區中進行提取來繼續滿足您的請求。為達到所需的容量，您可能會從超過您指定的集區 N 數量中接收 Spot 執行個體。同樣地，如果大多數集區沒有 Spot 容量，則您可能會從少於您指定的集區 N 數量中接收完整的所需容量。

Note

如果設定 Spot 執行個體為在開啟 [AMD SEV-SNP](#) 的情形下啟動，則需要支付額外的小時用量費，相當於所選執行個體類型 [隨需小時費率](#) 的 10%。如果配置策略使用價格作為輸入，則 Amazon EC2 Auto Scaling 不會包含此額外費用；僅使用 Spot 價格。

隨需執行個體

Amazon EC2 Auto Scaling 提供以下可用於隨需執行個體的分配策略：

lowest-price

Amazon EC2 Auto Scaling 根據當前的隨需價格自動在每個可用區域中部署最便宜的執行個體類型。

為滿足您所需的容量，您可能會在每個可用區域中收到多個執行個體類型的隨需執行個體。這取決於您請求多少容量。

prioritized

滿足隨需容量時，Amazon EC2 Auto Scaling 會根據啟動範本覆寫清單中的執行個體類型順序，決定先使用哪一種執行個體類型。例如，假設您按以下順序指定三個啟動範本覆寫：c5.large、c4.large 和 c3.large。當您的隨需執行個體啟動時，Auto Scaling 群組會按以下順序來滿足隨需容量：c5.large，c4.large，然後是 c3.large。

管理隨需執行個體的優先順序時，請考慮下列事項：

- 您可以預先支付使用量費用，使用 Savings Plans 或預留執行個體取得隨需執行個體的顯著折扣。如需詳細資訊，請參閱 [Amazon EC2 定價](#) 頁面。
- 對於預留執行個體，如果 Amazon EC2 Auto Scaling 啟動相符的執行個體類型，則適用於一般隨需執行個體定價的折扣費率。因此，如果您有適用於 c4.large 的未使用預留執行個體，您可以

設定執行個體類型優先順序，將預留執行個體類型的最高優先順序指定給某個 c4.large 執行個體類型。當 c4.large 執行個體啟動時，您可享受預留執行個體的定價。

- 透過 Savings Plans，使用 Amazon EC2 Instance Savings Plans 或 Compute Savings Plans 時，會套用一般隨需執行個體定價的折扣費率。透過 Savings Plans，在排定執行個體類型的優先順序時可享受更大的靈活性。只要您使用 Savings Plans 涵蓋的執行個體類型，就可以依任何優先順序進行設定。您也可以偶爾變更執行個體類型的整個順序，同時仍然會收到 Savings Plans 折扣費率。如需 Savings Plans 的詳細資訊，請參閱 [Savings Plans 使用者指南](#)。

配置策略如何使用權重

當您在覆寫 ("DesiredCapacityType": "vcpu"或在群組層級) "DesiredCapacityType": "memory-mib" 中指定 WeightedCapacity 參數時，配置策略的運作方式會與其他 Auto Scaling 群組相同。

假設您有一個 Auto Scaling 群組，其執行個體類型具有不同數量的 vCPUs。而您使用 lowest-price 作為 Spot 和隨需分配策略。如果您選擇根據每個執行個體類型的 vCPU 計數指派權重，則在履行時，Amazon EC2 Auto Scaling 將根據您所指派的權重值，啟動具有最低價格 (例如，每 vCPU) 的執行個體類型。如果是 Spot 執行個體，則意味著每個 vCPU 的最低 Spot 價格。如果是隨需執行個體，則意味著每個 vCPU 的最低隨需價格。

如需詳細資訊，請參閱[設定 Auto Scaling 群組以使用執行個體權重](#)。

使用屬性型執行個體類型選取項目建立混合執行個體群組

您可以指定一組描述運算需求的執行個體屬性，而不必手動選擇混合執行個體群組的執行個體類型。Amazon EC2 Auto Scaling 啟動執行個體時，Auto Scaling 群組使用的任何執行個體類型都必須與您所需的執行個體屬性相符。這就是所謂的屬性型執行個體類型選取範圍。

此方法非常適合那些可以靈活處理其所使用的執行個體類型的工作負載和架構，例如容器、大數據和 CI/CD。

屬性型執行個體類型選取範圍具有以下優勢：

- Spot 執行個體的最佳彈性 – Amazon EC2 Auto Scaling 可以從各種執行個體類型中選擇，以啟動 Spot 執行個體。這符合靈活處理執行個體類型的 Spot 最佳實務，讓 Amazon EC2 Spot 服務有更好的機會尋找和分配您所需的運算容量。
- 輕鬆使用正確的執行個體類型：由於有太多可用的執行個體類型，為您的工作負載尋找適當的執行個體類型可能會非常耗時。當您指定執行個體類型屬性時，執行個體類型會自動具有工作負載所需的屬性。

- 自動使用新的執行個體類型 – Auto Scaling 群組可以在發行時使用較新一代的執行個體類型。當新一代執行個體類型符合您的需求並與您為 Auto Scaling 群組選擇的分配策略保持一致時，系統會自動使用這些執行個體類型。

主題

- [屬性型執行個體類型選擇的運作方式](#)
- [價格保護](#)
- [效能保護](#)
- [先決條件](#)
- [使用屬性型執行個體類型選取項目（主控台）建立混合執行個體群組](#)
- [使用屬性型執行個體類型選取項目建立混合執行個體群組 \(AWS CLI\)](#)
- [範例組態](#)
- [預覽執行個體類型](#)
- [相關資源](#)

屬性型執行個體類型選擇的運作方式

透過屬性型執行個體類型選擇，您可以提供執行個體所需的執行個體屬性清單，例如：

- vCPU 數量：每個執行個體的 vCPU 數量下限和上限。
- 記憶體：每個執行個體的記憶體 GiB 下限和上限。
- 本機儲存：是否使用本機儲存的 EBS 或執行個體儲存體磁碟區。
- 爆量效能：是否使用 T 執行個體系列，包括 T4g、T3a、T3 和 T2 類型。

有許多選項可用於定義您的執行個體需求。如需每個選項和預設值的說明，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [InstanceRequirements](#)。

當您的 Auto Scaling 群組需要啟動執行個體時，它會搜尋符合您指定屬性且可在該可用區域中使用的執行個體類型。然後，配置策略會決定要啟動哪些相符的執行個體類型。根據預設，屬性型執行個體類型選擇已啟用價格保護功能，以防止 Auto Scaling 群組啟動超過預算閾值的執行個體類型。

根據預設，您在設定 Auto Scaling 群組的所需容量時，會使用執行個體數量做為度量單位，這表示每個執行個體都會計為一個單位。

或者，您可以將所需容量的值設定為 vCPU 的數量或記憶體量。若要這樣做，請使用 中的所需容量類型下拉式清單欄位，AWS 管理主控台 或 `CreateAutoScalingGroup` 或 `UpdateAutoScalingGroup` API 操作中的 `DesiredCapacityType` 屬性。然後，Amazon EC2 Auto Scaling 會啟動滿足所需 vCPU 或記憶體容量所需的執行個體數量。例如，如果您使用 vCPUs 做為所需的容量類型，並使用各具有 2 個 vCPUs 執行個體，則 10 個 vCPUs 的所需容量會啟動 5 個執行個體。這是一個可以替代[執行個體加權](#)的有用方案。

價格保護

透過價格保護，您可以指定您願意為 Auto Scaling 群組啟動的 EC2 執行個體支付的最高價格。價格保護功能可防止 Auto Scaling 群組使用您認為過於昂貴的執行個體類型，即使它們剛好符合您指定的屬性。

價格保護預設為啟用，並具有個別的隨需執行個體和 Spot 執行個體價格閾值。當 Amazon EC2 Auto Scaling 需要啟動新的執行個體時，不會啟動任何價格高於相關閾值的執行個體類型。

主題

- [隨需價格保護](#)
- [Spot 價格保護](#)
- [自訂價格保護](#)

隨需價格保護

對於隨需執行個體，您可以將您願意支付的最高隨需價格定義為高於已識別隨需價格的百分比。識別的隨需價格是價格最低的目前世代 C、M 或 R 執行個體類型，以及您指定的屬性。

如果未明確定義隨需價格保護值，則會使用高於指定隨需價格 20% 的預設隨需價格上限。

Spot 價格保護

根據預設，Amazon EC2 Auto Scaling 會自動套用最佳 Spot 執行個體價格保護，以一致地從各種執行個體類型中選擇。您也可以自行手動設定價格保護。不過，讓 Amazon EC2 Auto Scaling 為您執行此操作，您可以提高 Spot 容量履行的可能性。

您可以使用下列其中一個選項手動指定價格保護。如果您手動設定價格保護，我們建議您使用第一個選項。

- 已識別的隨需價格百分比 – 已識別的隨需價格是具有您指定屬性之價格最低的目前世代 C、M 或 R 執行個體類型的價格。

- 高於已識別 Spot 價格的百分比 – 已識別的 Spot 價格是具有您指定屬性的最低價格目前世代 C、M 或 R 執行個體類型的價格。我們不建議使用此選項，因為 Spot 價格可能會波動，因此您的價格保護閾值也可能會波動。

自訂價格保護

您可以在 Amazon EC2 Auto Scaling 主控台或使用 AWS CLI 或 SDKs 自訂價格保護閾值。

- 在 主控台中，使用其他執行個體屬性中的隨需價格保護和 Spot 價格保護設定。
- 在 [InstanceRequirements](#) 結構中，若要指定隨需執行個體價格保護閾值，請使用 `OnDemandMaxPricePercentageOverLowestPrice` 屬性。若要指定 Spot 執行個體價格保護閾值，請使用 `MaxSpotPriceAsPercentageOfOptimalOnDemandPrice` 或 `SpotMaxPricePercentageOverLowestPrice` 屬性。

如果您將所需容量類型 (`DesiredCapacityType`) 設定為 vCPUs 或記憶體 GiB，則價格保護會根據每個 vCPU 或每個記憶體價格套用，而不是每個執行個體的價格。

您也可以關閉價格保護。若要表示沒有價格保護閾值，請指定高百分比值，例如 999999。

Note

如果沒有最新一代的 C、M 或 R 執行個體類型符合您指定的屬性，則價格保護仍然適用。找不到相符項目時，識別的價格來自價格最低的目前世代執行個體類型，或不符合價格最低的前一代執行個體類型，以符合您的屬性。

效能保護

效能保護功能可確保 Auto Scaling 群組使用類似或超過指定效能基準的執行個體類型。若要使用效能保護，您可以將執行個體系列指定為基準參考。指定執行個體系列的功能會建立最低可接受效能層級。當 Auto Scaling 選取執行個體類型時，它會考慮您指定的屬性和效能基準。低於效能基準的執行個體類型，將自動排除在選取範圍之外，即使這些類型符合您的其他指定屬性。這可確保所有選取的執行個體類型所提供的效能，與指定執行個體系列所建立的基準類似或更好。Auto Scaling 使用此基準來引導執行個體類型選擇，但無法保證選取的執行個體類型一律會超過每個應用程式的基準。

目前，此功能僅支援以 CPU 效能作為基準效能因素。指定執行個體系列的 CPU 效能做為效能基準，確保選取的執行個體類型類似於或超過此基準。具有相同 CPU 處理器的執行個體系列會產生相同的篩

選結果，即使其網路或磁碟效能不同。例如，指定 c6in 或 c6i 作為基準參考，會產生相同的效能型篩選結果，因為兩個執行個體系列都使用相同的 CPU 處理器。

不支援的執行個體系列

效能保護不支援下列執行個體系列：

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

如果您透過指定支援的執行個體系列來啟用效能保護，則傳回的執行個體類型將排除上述不支援的執行個體系列。

範例：設定 CPU 效能基準

在下面的示例中，執行個體需求是使用執行個體類型啟動，該執行個體類型有與 c6i 執行個體系列相同的 CPU 核心性能。這將篩選效能較低的 CPU 處理器的執行個體類型，即使它們符合您的其他指定執行個體需求，例如 vCPU 數量。例如，如果您指定的執行個體屬性包含 4 個 vCPU 和 16GB 的記憶體，則具有這些屬性但 CPU 效能低於 c6i 的執行個體類型將被排除在選擇之外。

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

考量事項

使用效能保護時，請考慮下列事項：

- 您可以指定執行個體類型或執行個體屬性，但不能同時指定兩者。
- 您最多可以在請求組態中指定四個 InstanceRequirements 結構。

先決條件

- 建立啟動範本。如需詳細資訊，請參閱 [為 Auto Scaling 群組建立啟動範本](#)。
- 驗證啟動範本是否尚未請求 Spot 執行個體。

使用屬性型執行個體類型選取項目（主控台）建立混合執行個體群組

請遵循以下程序，使用屬性型執行個體類型選取範圍來建立混合執行個體群組。為了協助您有效率地完成這些步驟，我們會略過某些選用的區段。

對於大多數一般用途的工作負載來說，只需指定所需的 vCPU 和記憶體數量就已足夠。對於進階使用案例，您可以指定儲存類型、網路介面、CPU 製造商和加速器類型等屬性。

若要檢閱混合執行個體群組的最佳實務，請參閱 [建立混合執行個體群組的設定概觀](#)。

建立混合執行個體群組

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在畫面上方的導覽列中，選擇您用來建立啟動範本的相同 AWS 區域。
3. 選擇建立 Auto Scaling 群組。
4. 在選擇啟動範本或組態頁面上，針對 Auto Scaling 群組名稱，請輸入 Auto Scaling 群組的名稱。
5. 若要選擇您的啟動範本，請執行以下操作：
 - a. 針對啟動範本，選擇現有的啟動範本。
 - b. 針對 Launch template version (啟動範本版本)，選擇 Auto Scaling 群組擴增時使用啟動範本的預設、最新或特定版本。
 - c. 確認您的啟動範本支援您打算使用的所有選項，然後選擇 Next (下一步)。
6. 在選擇執行個體啟動選項頁面上，請執行以下操作：
 - a. 對於 Instance type requirements (執行個體類型需求)，選擇 Override launch template (覆寫啟動範本)。

 Note

如果您選擇的啟動範本已包含一組執行個體屬性 (例如 vCPU 和記憶體)，則會顯示執行個體屬性。這些屬性會新增至 Auto Scaling 群組屬性，您隨時可從 Amazon EC2 Auto Scaling 主控台更新這些屬性。

- b. 在 Specify instance attributes (指定執行個體屬性) 下，首先輸入您的 vCPU 和記憶體需求。
 - 對於 vCPUs，輸入所需的 vCPU 數量下限和上限。若要指定無限制，請選取 No minimum (無下限)、No maximum (無上限)，或兩者。
 - 對於 Memory (GiB) (記憶體 (GiB))，輸入所需記憶體數量的下限和上限。若要指定無限制，請選取 No minimum (無下限)、No maximum (無上限)，或兩者。
- c. (選用) 對於 Additional instance attributes (其他執行個體屬性)，您可以選擇性地指定一或多個屬性，以更詳細地表達您的運算需求。每個額外屬性都會將進一步的限制新增至您的請求。
- d. 展開預覽比對執行個體類型，以檢視具有您指定屬性的執行個體類型。
- e. 在執行個體購買選項下的執行個體分佈，分別指定群組要以隨需執行個體和 Spot 執行個體啟動的百分比。如果您的應用程式無狀態、可容錯並且可以處理被中斷的執行個體，則您可以指定較高的 Spot 執行個體百分比。
- f. (選用) 當您指定 Spot 執行個體的百分比時，您可以勾選包含隨需基本容量旁的核取方塊，然後指定 Auto Scaling 群組初始容量中應以隨需執行個體履行的數量下限。超過基本容量的項目會使用 Instances distribution (執行個體分佈) 設定來決定啟動多少隨需執行個體和 Spot 執行個體。
- g. 在 Allocation strategies (分配策略) 下，On-Demand allocation strategy (隨需分配策略) 會自動選取 Lowest price (最低價格)，且無法對其進行變更。
- h. 對於 Spot allocation strategy (Spot 分配策略)，選擇分配策略。依預設會選取最佳化的價格容量。
- i. 對於容量重新平衡，請選擇啟用或停用容量重新平衡。當 Spot 執行個體因 Spot 中斷而即將終止時，可使用容量重新平衡來自動回應。如需詳細資訊，請參閱[Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體](#)。
- j. 在 Network (網路) 下，對於 VPC，選擇 VPC。Auto Scaling 群組必須與您在啟動範本中指定的安全群組建立在同一個 VPC 中。
- k. 對於可用區域和子網路，請選擇指定 VPC 中的一個或多個子網路。在多個可用區域中，請使用子網路，以獲得高可用性。如需詳細資訊，請參閱[選擇 VPC 子網路時的注意事項](#)。
- l. 選擇下一步、下一步。

7. 針對設定群組大小和擴展政策步驟，請執行以下操作：
 - a. 若要以執行個體以外的單位來量測所需容量，應為群組大小、所需容量類型選擇適當的選項。支援 Units (單位)、vCPU 和 Memory GiB (記憶體 GiB)。根據預設，Amazon EC2 Auto Scaling 會指定 Units (單位)，其會轉譯為執行個體數目。
 - b. 所需容量為 Auto Scaling 群組的初始大小。
 - c. 在擴展區段的擴展限制下，如果所需容量的新值大於所需容量下限和所需容量上限，則所需容量上限會自動增加到新的所需容量值。您可視需要變更這些限制。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
8. 選擇 Skip to review (跳過以進行檢閱)。
9. 在 Review (檢閱) 頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

使用屬性型執行個體類型選取項目建立混合執行個體群組 (AWS CLI)

使用命令列建立混合執行個體群組

請使用以下其中一個命令：

- [create-auto-scaling-group](#) (AWS CLI)
- [New-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

範例組態

若要使用 AWS CLI 建立具有屬性型執行個體類型選取範圍的 Auto Scaling 群組，請使用以下 [create-auto-scaling-group](#) 命令。

會指定以下執行個體屬性：

- VCpuCount – 執行個體類型必須具有最少 4 個，最多 8 個的 vCPU。
- MemoryMiB – 執行個體類型必須具有最少 16,384 MiB 的記憶體。
- CpuManufacturers – 執行個體類型必須具有 Intel 製造的 CPU。

JSON

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

以下是範例 config.json 檔案。


```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredCapacityType": "units",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Default"
      },
      "Overrides": [{
        "InstanceRequirements": {
          "VCpuCount": {"Min": 4, "Max": 8},
          "MemoryMiB": {"Min": 16384},
          "CpuManufacturers": ["intel"]
        }
      }]
    },
    "InstancesDistribution": {
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "price-capacity-optimized"
    }
  },
  "MinSize": 0,
  "MaxSize": 100,
  "DesiredCapacity": 4,
  "DesiredCapacityType": "units",
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

要將所需容量的值設定為 vCPU 的數量或記憶體容量，請在檔案中指定 "DesiredCapacityType": "vcpu" 或 "DesiredCapacityType": "memory-mib"。預設所需的容量類型為 units，它將所需容量的值設定為執行個體數量。

YAML

或者，您可使用以下 [create-auto-scaling-group](#) 命令建立 Auto Scaling 群組。這將參考 YAML 檔案作為 Auto Scaling 群組的唯一參數。

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

以下是範例 config.yaml 檔案。

```

---
AutoScalingGroupName: my-asg
DesiredCapacityType: units
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceRequirements:
          VCpuCount:
            Min: 2
            Max: 4
          MemoryMiB:
            Min: 2048
          CpuManufacturers:
            - intel
      InstancesDistribution:
        OnDemandPercentageAboveBaseCapacity: 50
        SpotAllocationStrategy: price-capacity-optimized
  MinSize: 0
  MaxSize: 100
  DesiredCapacity: 4
  DesiredCapacityType: units
  VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782

```

要將所需容量的值設定為 vCPU 的數量或記憶體容量，請在檔案中指定 `DesiredCapacityType: vcpu` 或 `DesiredCapacityType: memory-mib`。預設所需的容量類型為 `units`，它將所需容量的值設定為執行個體數量。

如需如何搭配屬性型執行個體類型選擇使用多個啟動範本的範例，請參閱 [使用多個啟動範本](#)。

預覽執行個體類型

您可以預覽符合運算需求的執行個體類型，而無需啟動它們，並根據需要調整您的需求。在 Amazon EC2 Auto Scaling 主控台中建立 Auto Scaling 群組時，執行個體類型的預覽會顯示在 Choose instance launch options (選擇執行個體啟動選項) 頁面的 Preview matching instance types (預覽相符執行個體類型) 中。

或者，您可以使用 AWS CLI 或 SDK 進行 Amazon EC2

[GetInstanceTypesFromInstanceRequirements](#) API 呼叫來預覽執行個體類型。傳遞請求中的

InstanceRequirements 參數，使用您會用於建立或更新 Auto Scaling 群組的相同格式。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[使用指定屬性預覽執行個體類型](#)。

相關資源

若要進一步了解屬性型執行個體類型選擇，請參閱 AWS 部落格上的 [EC2 Auto Scaling 和 EC2 機群的屬性型執行個體類型選擇](#)。

使用 CloudFormation 建立 Auto Scaling 群組時，您可以宣告屬性型執行個體類型選取範圍。如需詳細資訊，請參閱《CloudFormation 使用者指南》中的 [Auto scaling 範本程式碼片段](#) 一節。

手動選擇執行個體類型以建立混合執行個體群組

本主題示範如何透過手動選擇執行個體類型在單一 Auto Scaling 群組中啟動多個執行個體類型。

若您偏好以執行個體屬性作為選取執行個體類型的條件，請參閱 [使用屬性型執行個體類型選取項目建立混合執行個體群組](#)。

目錄

- [先決條件](#)
- [建立混合執行個體群組 \(主控台\)](#)
- [建立混合執行個體群組 \(AWS CLI\)](#)
- [範例組態](#)

先決條件

- 建立啟動範本。如需詳細資訊，請參閱 [為 Auto Scaling 群組建立啟動範本](#)。
- 驗證啟動範本是否尚未請求 Spot 執行個體。

建立混合執行個體群組 (主控台)

請使用下列程序，透過手動選擇您的群組可以啟動的執行個體類型來建立混合執行個體群組。為了協助您有效率地完成這些步驟，我們會略過某些選用的區段。

若要檢閱混合執行個體群組的最佳實務，請參閱 [建立混合執行個體群組的設定概觀](#)。

建立混合執行個體群組

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。

2. 在畫面上方的導覽列中，選擇您用來建立啟動範本的相同 AWS 區域。
3. 選擇建立 Auto Scaling 群組。
4. 在選擇啟動範本或組態頁面上，針對 Auto Scaling 群組名稱，請輸入 Auto Scaling 群組的名稱。
5. 若要選擇您的啟動範本，請執行以下操作：
 - a. 針對啟動範本，選擇現有的啟動範本。
 - b. 針對啟動範本版本，選擇 Auto Scaling 群組擴增時使用啟動範本的預設、最新或特定版本。
 - c. 確認您的啟動範本支援您打算使用的所有選項，然後選擇下一步。
6. 在選擇執行個體啟動選項頁面中，執行以下操作：
 - a. 對於 Instance type requirements (執行個體類型需求)，選擇 Override launch template (覆寫啟動範本)，然後選擇 Manually add instance types (手動新增執行個體類型)。
 - b. 選擇您的執行個體類型。您可以使用我們的建議作為起點。依預設選取系列和層代靈活。
 - 若要變更執行個體類型的順序，請使用箭頭。如果您選擇支援優先順序的分配策略，則執行個體類型順序會設定其啟動優先順序。
 - 若要移除執行個體類型，請選擇 X。
 - (選用) 對於權重欄中的方塊，為每個執行個體類型指定相對權重。若要這麼做，請輸入該類型的執行個體計入群組所需容量的單位數。如果執行個體類型提供不同的 vCPU、記憶體、儲存或網路頻寬功能，則這麼做可能會很實用。如需詳細資訊，請參閱[設定 Auto Scaling 群組以使用執行個體權重](#)。
 - c. 在 Instance purchase options (執行個體購買選項) 下，針對 Instances distribution (執行個體分佈)，分別指定群組要以隨需執行個體和 Spot 執行個體啟動的百分比。如果您的應用程式無狀態、可容錯並且可以處理被中斷的執行個體，則您可以指定較高的 Spot 執行個體百分比。
 - d. (選用) 當您指定 Spot 執行個體的百分比時，您可以勾選包含隨需基本容量旁的核取方塊，然後指定 Auto Scaling 群組初始容量中應以隨需執行個體履行的數量下限。超過基本容量的項目會使用執行個體分佈設定來決定啟動多少隨需執行個體和 Spot 執行個體。
 - e. 在 Allocation strategies (分配策略) 下，對於 On-Demand allocation strategy (隨需分配策略)，請選擇分配策略。手動選擇執行個體類型時，依預設會選取 Prioritized (按優先順序排序)。
 - f. 對於 Spot allocation strategy (Spot 分配策略)，選擇分配策略。依預設會選取最佳化的價格容量。

如果您選擇容量最佳化，則可以選擇性地勾選排定執行個體類型優先順序方塊，讓 Amazon EC2 Auto Scaling 根據執行個體類型的列出順序，選擇要先啟動的執行個體類型。

- g. 對於容量重新平衡，請選擇啟用或停用容量重新平衡。當 Spot 執行個體因 Spot 中斷而即將終止時，可使用容量重新平衡來自動回應。如需詳細資訊，請參閱[Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體](#)。
 - h. 在 Network (網路) 下，對於 VPC，選擇 VPC。Auto Scaling 群組必須與您在啟動範本中指定的安全群組建立在同一個 VPC 中。
 - i. 對於可用區域和子網路，請選擇指定 VPC 中的一個或多個子網路。在多個可用區域中，請使用子網路，以獲得高可用性。如需詳細資訊，請參閱[選擇 VPC 子網路時的注意事項](#)。
 - j. 選擇下一步、下一步。
7. 針對設定群組大小和擴展政策步驟，請執行以下操作：
- a. 對於所需容量，在群組大小下輸入要啟動之執行個體的初始數量。

依預設，所需容量表示為執行個體數量。如果您為執行個體類型指派權重，則必須將該值轉換為與指派權重相同的測量單位，例如 vCPU 數量。
 - b. 在擴展區段的擴展限制下，如果所需容量的新值大於所需容量下限和所需容量上限，則所需容量上限會自動增加到新的所需容量值。您可視需要變更這些限制。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
8. 選擇 Skip to review (跳過以進行檢閱)。
9. 在 Review (檢閱) 頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

建立混合執行個體群組 (AWS CLI)

使用命令列建立混合執行個體群組

請使用以下其中一個命令：

- [create-auto-scaling-group](#) (AWS CLI)
- [New-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

範例組態

下列範例組態示範如何使用不同 Spot 分配策略建立混合執行個體群組。

 Note

這些範例展示如何使用 JSON 或 YAML 格式組態檔案。如果您使用 AWS CLI 版本 1，則必須指定 JSON 格式的組態檔案。如果您使用 AWS CLI 版本 2，您可以指定 YAML 或 JSON 格式的組態檔案。

範例

- [範例 1：使用 capacity-optimized 配置策略啟動 Spot 執行個體](#)
- [範例 2：使用 capacity-optimized-prioritized 配置策略啟動 Spot 執行個體](#)
- [範例 3：使用分散在兩個集區的 lowest-price 配置策略啟動 Spot 執行個體](#)
- [範例 4：使用 price-capacity-optimized 分配策略啟動 競價型執行個體](#)

範例 1：使用 **capacity-optimized** 配置策略啟動 Spot 執行個體

下列 [create-auto-scaling-group](#) 命令會建立 Auto Scaling 群組，其會指定下列項目：

- 要啟動多少百分比的群組來做為隨需執行個體 (0)，以及隨需執行個體要開始的基本數量 (1)。
- 按照優先順序啟動的執行個體類型
(c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.large)
- 在其中啟動執行個體的子網路 (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)。
每個對應於不同的可用區域。
- 啟動範本 (my-launch-template) 和啟動範本的版本 (\$Default)。

Amazon EC2 Auto Scaling 在嘗試滿足您的隨需容量時，會先啟動 c5.large 執行個體類型。Spot 執行個體根據 Spot 執行個體容量，來自於每個可用區域的最佳 Spot 集區。

JSON

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

config.json 檔案包含下列內容。

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
```

```

    "LaunchTemplateSpecification": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "$Default"
    },
    "Overrides": [
      {
        "InstanceType": "c5.large"
      },
      {
        "InstanceType": "c5a.large"
      },
      {
        "InstanceType": "m5.large"
      },
      {
        "InstanceType": "m5a.large"
      },
      {
        "InstanceType": "c4.large"
      },
      {
        "InstanceType": "m4.large"
      },
      {
        "InstanceType": "c3.large"
      },
      {
        "InstanceType": "m3.large"
      }
    ]
  },
  "InstancesDistribution": {
    "OnDemandBaseCapacity": 1,
    "OnDemandPercentageAboveBaseCapacity": 0,
    "SpotAllocationStrategy": "capacity-optimized"
  }
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```


YAML

或者，您可使用以下 [create-auto-scaling-group](#) 命令建立 Auto Scaling 群組。這將參考 YAML 檔案作為 Auto Scaling 群組的唯一參數。

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

config.yaml 檔案包含下列內容。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large
      - InstanceType: c3.large
      - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

範例 2：使用 `capacity-optimized-prioritized` 配置策略啟動 Spot 執行個體

下列 [create-auto-scaling-group](#) 命令會建立 Auto Scaling 群組，其會指定下列項目：

- 要啟動多少百分比的群組來做為隨需執行個體 (0)，以及隨需執行個體要開始的基本數量 (1)。
- 按照優先順序啟動的執行個體類型
(c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.large)

- 在其中啟動執行個體的子網路 (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)。每個對應於不同的可用區域。
- 啟動範本 (my-launch-template) 和啟動範本的版本 (\$Latest)。

Amazon EC2 Auto Scaling 在嘗試滿足您的隨需容量時，會先啟動 c5.large 執行個體類型。Amazon EC2 Auto Scaling 在嘗試滿足 Spot 容量時，會竭盡全力遵循執行個體類型優先順序。不過，它會先針對容量進行最佳化。

JSON

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

config.json 檔案包含下列內容。

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
      "Overrides": [
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        },
        {
          "InstanceType": "m5.large"
        },
        {
          "InstanceType": "m5a.large"
        },
        {
          "InstanceType": "c4.large"
        },
        {
          "InstanceType": "m4.large"
        }
      ]
    }
  }
}
```

```

        {
            "InstanceType": "c3.large"
        },
        {
            "InstanceType": "m3.large"
        }
    ]
},
"InstancesDistribution": {
    "OnDemandBaseCapacity": 1,
    "OnDemandPercentageAboveBaseCapacity": 0,
    "SpotAllocationStrategy": "capacity-optimized-prioritized"
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```

YAML

或者，您可使用以下 [create-auto-scaling-group](#) 命令建立 Auto Scaling 群組。這將參考 YAML 檔案作為 Auto Scaling 群組的唯一參數。

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

config.yaml 檔案包含下列內容。

```

---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large

```

```

- InstanceType: c3.large
- InstanceType: m3.large
InstancesDistribution:
  OnDemandBaseCapacity: 1
  OnDemandPercentageAboveBaseCapacity: 0
  SpotAllocationStrategy: capacity-optimized-prioritized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782

```

範例 3：使用分散在兩個集區的 **lowest-price** 配置策略啟動 Spot 執行個體

下列 [create-auto-scaling-group](#) 命令會建立 Auto Scaling 群組，其會指定下列項目：

- 要以隨需執行個體形式啟動的群組百分比 (50)。(這不會指定開始的隨需執行個體的基本數目。)
- 按照優先順序啟動的執行個體類型
(c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.large)
- 在其中啟動執行個體的子網路 (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)。每個對應於不同的可用區域。
- 啟動範本 (my-launch-template) 和啟動範本的版本 (\$Latest)。

Amazon EC2 Auto Scaling 在嘗試滿足您的隨需容量時，會先啟動 c5.large 執行個體類型。就 Spot 容量而言，Amazon EC2 Auto Scaling 嘗試在各可用區域的兩個最低價格的集區中平均啟動 Spot 執行個體。

JSON

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

config.json 檔案包含下列內容。

```

{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      }
    }
  },

```

```

    "Overrides": [
      {
        "InstanceType": "c5.large"
      },
      {
        "InstanceType": "c5a.large"
      },
      {
        "InstanceType": "m5.large"
      },
      {
        "InstanceType": "m5a.large"
      },
      {
        "InstanceType": "c4.large"
      },
      {
        "InstanceType": "m4.large"
      },
      {
        "InstanceType": "c3.large"
      },
      {
        "InstanceType": "m3.large"
      }
    ],
    "InstancesDistribution": {
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "lowest-price",
      "SpotInstancePools": 2
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
  }

```

YAML

或者，您可使用以下 [create-auto-scaling-group](#) 命令建立 Auto Scaling 群組。這將參考 YAML 檔案作為 Auto Scaling 群組的唯一參數。

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

config.yaml 檔案包含下列內容。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large
      - InstanceType: c3.large
      - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: lowest-price
    SpotInstancePools: 2
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

範例 4：使用 **price-capacity-optimized** 分配策略啟動 競價型執行個體

下列 [create-auto-scaling-group](#) 命令會建立 Auto Scaling 群組，其會指定下列項目：

- 要以隨需執行個體形式啟動的群組百分比 (30)。(這不會指定開始的隨需執行個體的基本數目。)
- 按照優先順序啟動的執行個體類型
(c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.large)
- 在其中啟動執行個體的子網路 (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)。
每個對應於不同的可用區域。
- 啟動範本 (my-launch-template) 和啟動範本的版本 (\$Latest)。

Amazon EC2 Auto Scaling 在嘗試滿足您的隨需容量時，會先啟動 `c5.large` 執行個體類型。對於 Spot 容量，Amazon EC2 Auto Scaling 嘗試從 Spot 執行個體集區以可能的最低價格啟動 Spot 執行個體，但同時也具有正啟動的執行個體數量的最佳容量。

JSON

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

`config.json` 檔案包含下列內容。

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
      "Overrides": [
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        },
        {
          "InstanceType": "m5.large"
        },
        {
          "InstanceType": "m5a.large"
        },
        {
          "InstanceType": "c4.large"
        },
        {
          "InstanceType": "m4.large"
        },
        {
          "InstanceType": "c3.large"
        },
        {
          "InstanceType": "m3.large"
        }
      ]
    }
  }
}
```



```

    ]
  },
  "InstancesDistribution": {
    "OnDemandPercentageAboveBaseCapacity": 30,
    "SpotAllocationStrategy": "price-capacity-optimized"
  }
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```

YAML

或者，您可使用以下 [create-auto-scaling-group](#) 命令建立 Auto Scaling 群組。這將參考 YAML 檔案作為 Auto Scaling 群組的唯一參數。

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

config.yaml 檔案包含下列內容。

```

---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large
      - InstanceType: c3.large
      - InstanceType: m3.large
    InstancesDistribution:
      OnDemandPercentageAboveBaseCapacity: 30
      SpotAllocationStrategy: price-capacity-optimized
  MinSize: 1

```

```
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

設定 Auto Scaling 群組以使用執行個體權重

當您使用多個執行個體類型時，您可以指定要與每個執行個體類型建立關聯的單位數量，然後使用相同的度量單位來指定群組的容量。此容量規格選項稱為權重。

例如，假設您執行運算密集型應用程式，該應用程式至少要 8 個 vCPU 和 15 GiB 的 RAM 來執行最佳效能。如果您使用 c5.2xlarge 做為基本單位，則下列任何 EC2 執行個體類型都可以滿足您的應用程式需求。

執行個體類型範例

執行個體類型	vCPU	記憶體 (GiB)
c5.2xlarge	8	16
c5.4xlarge	16	32
c5.12xlarge	48	96
c5.18xlarge	72	144
c5.24xlarge	96	192

預設情況下，所有執行個體類型無論大小為何，都具有相同的權重。換句話說，無論 Amazon EC2 Auto Scaling 啟動的是大型或小型執行個體類型，每個執行個體都會相同地計入 Auto Scaling 群組的所需容量。

不過，透過權重，您可以指派數值來指定要與每個執行個體類型建立關聯的單位數量。例如，如果執行個體大小不同，c5.2xlarge 執行個體的權重可能為 2，而 c5.4xlarge (兩倍大) 的權重可能為 4，依此類推。然後，當 Amazon EC2 Auto Scaling 擴展群組時，這些權重就會轉換為每個執行個體計入所需容量的單位數量。

權重不會改變 Amazon EC2 Auto Scaling 選擇啟動的執行個體類型，而會改為配置策略執行此動作。如需詳細資訊，請參閱[多種執行個體類型的配置策略](#)。

⚠ Important

若要設定 Auto Scaling 群組以使用 vCPU 數量或每個執行個體類型的記憶體量來滿足所需容量，建議您選擇屬性型執行個體類型。設定 `DesiredCapacityType` 參數會根據您為此參數設定的值，自動指定要與每個執行個體類型建立關聯的單位數量。如需詳細資訊，請參閱[使用屬性型執行個體類型選取項目建立混合執行個體群組](#)。

目錄

- [考量事項](#)
- [執行個體權重行為](#)
- [設定 Auto Scaling 群組以使用權重](#)
- [每單位小時 Spot 價格範例](#)

考量事項

本節討論有效實作權重的重要考量。

- 選擇一些符合您應用程式效能需求的執行個體類型。根據 Auto Scaling 群組的功能，決定每個執行個體類型應該計入所需容量的權重。這些權重適用於目前和未來的執行個體。
- 避免權重之間的較大範圍。例如，當下一個較大的執行個體類型具有 200 的權重時，請勿為執行個體類型指定權重 1。最小權重和最大權重之間的差異也不應過於極端。極端權重差異可能會對成本效能最佳化產生負面影響。
- 以單位指定群組所需的容量，而非執行個體。例如，如果您使用 vCPU 型權重，請設定所需的核芯數量，以及最小值和最大值。
- 設定您的權重和所需的容量，使所需的容量至少比您最大權重大兩到三倍。

更新現有群組時，請注意下列事項：

- 當您將權重新增至現有群組時，請包含目前使用中所有執行個體類型的權重。
- 當您新增或變更權重時，Amazon EC2 Auto Scaling 會根據新的權重值啟動或終止執行個體，以達到所需的容量。
- 如果您移除執行個體類型，即使不再定義，執行該類型的執行個體仍會保留其最後權重。

執行個體權重行為

當您使用執行個體權重時，Amazon EC2 Auto Scaling 的行為方式如下：

- 目前容量將達到所需的容量或高於它。如果啟動的執行個體超過剩餘的所需容量單位，則目前的容量可能會超過所需的容量。例如，假設您指定兩個執行個體類型 c5.2xlarge 和 c5.12xlarge，以及指派 c5.2xlarge 執行個體權重為 2 和 c5.12xlarge 為 12。如果剩餘五個單位可以滿足所需容量，並且 Amazon EC2 Auto Scaling 佈建的是 c5.12xlarge，則所需容量會超過七個單位。
- 啟動執行個體時，Amazon EC2 Auto Scaling 會優先分配容量到各個可用區域，並遵循分配策略，而不是超過所需的容量。
- Amazon EC2 Auto Scaling 可以使用您偏好的配置策略，超過容量上限，以維持可用區域之間的平衡。Amazon EC2 Auto Scaling 強制執行的硬性限制是您想要的容量加上最大權重。

設定 Auto Scaling 群組以使用權重

您可以設定 Auto Scaling 群組以使用權重，如下列 AWS CLI 範例所示。如需使用主控台的說明，請參閱「[手動選擇執行個體類型以建立混合執行個體群組](#)」。

設定新的 Auto Scaling 群組以使用權重 (AWS CLI)

使用 [create-auto-scaling-group](#) 命令。例如，以下命令會藉由指定下列項目，從而建立新的 Auto Scaling 群組並指派權重：

- 要以隨需執行個體形式啟動的群組百分比 (0)
- 每個可用區域中 Spot 執行個體的分配策略 (capacity-optimized)
- 按照優先順序啟動的執行個體類型 (m4.16xlarge、m5.24xlarge)
- 與執行個體類型 (16、24) 之間的相對大小差異 (vCPU) 對應的執行個體權重
- 要在其中啟動執行個體的子網路 (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)，每條子網路各自對應至不同的可用區域
- 啟動範本 (my-launch-template) 和啟動範本的版本 (\$Latest)

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

config.json 檔案包含下列內容。

```
{  
  "AutoScalingGroupName": "my-asg",
```

```

    "MixedInstancesPolicy": {
      "LaunchTemplate": {
        "LaunchTemplateSpecification": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "$Latest"
        },
        "Overrides": [
          {
            "InstanceType": "m4.16xlarge",
            "WeightedCapacity": "16"
          },
          {
            "InstanceType": "m5.24xlarge",
            "WeightedCapacity": "24"
          }
        ]
      },
      "InstancesDistribution": {
        "OnDemandPercentageAboveBaseCapacity": 0,
        "SpotAllocationStrategy": "capacity-optimized"
      }
    },
    "MinSize": 160,
    "MaxSize": 720,
    "DesiredCapacity": 480,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
  }
}

```

設定現有的 Auto Scaling 群組以使用權重 (AWS CLI)

使用 [update-auto-scaling-group](#) 命令。例如，下列命令透過指定下列項目，從而對現有 Auto Scaling 群組中的執行個體類型指派權重：

- 按照優先順序啟動的執行個體類型 (c5.18xlarge、c5.24xlarge、c5.2xlarge、c5.4xlarge)
- 與執行個體類型 (18、24、2、4) 之間的相對大小差異 (vCPU) 對應的執行個體權重
- 新的、已增加的所需的容量，大於最大的權重

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

config.json 檔案包含下列內容。

```
{
  "AutoScalingGroupName": "my-existing-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "Overrides": [
        {
          "InstanceType": "c5.18xlarge",
          "WeightedCapacity": "18"
        },
        {
          "InstanceType": "c5.24xlarge",
          "WeightedCapacity": "24"
        },
        {
          "InstanceType": "c5.2xlarge",
          "WeightedCapacity": "2"
        },
        {
          "InstanceType": "c5.4xlarge",
          "WeightedCapacity": "4"
        }
      ]
    }
  },
  "MinSize": 0,
  "MaxSize": 100,
  "DesiredCapacity": 100
}
```

使用命令列驗證權重

請使用以下其中一個命令：

- [describe-auto-scaling-groups](#) (AWS CLI)
- [Get-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

每單位小時 Spot 價格範例

下列表格將美國東部 (維吉尼亞北部) 不同可用區域中 Spot 執行個體每單位小時的價格，與相同區域中隨需執行個體的價格做比較。顯示的價格是範例的定價，而不是目前的定價。這些是您每個執行個體小時的成本。

範例：每個執行個體小時的 Spot 定價

執行個體類型	us-east-1a	us-east-1b	us-east-1c	隨需定價
c5.2xlarge	0.180 USD	0.191 USD	0.170 USD	0.34 USD
c5.4xlarge	0.341 USD	0.361 USD	0.318 USD	0.68 USD
c5.12xlarge	0.779 USD	0.777 USD	0.777 USD	2.04 USD
c5.18xlarge	1.207 USD	1.475 USD	1.357 USD	3.06 USD
c5.24xlarge	1.555 USD	1.555 USD	1.555 USD	4.08 USD

透過執行個體權重，您可以根據每單位小時使用量來評估成本。您可以透過將執行個體類型價格除以它所代表的單位數，來判斷每單位小時的價格。對於隨需執行個體，部署一個執行個體類型時的每單位小時價格與部署相同執行個體類型的不同大小價格相同。不過，每單位小時的 Spot 價格會因 Spot 集區而有所不同。

下列範例示範每單位小時競價型價格計算如何與執行個體權重搭配使用。為了方便計算，假設您只想在 us-east-1a 中啟動 Spot 執行個體。下表中擷取了每單位小時的價格。

範例：每單位小時 Spot 價格

執行個體類型	us-east-1a	執行個體權重	每個單位小時的價格
c5.2xlarge	0.180 USD	2	0.090 USD
c5.4xlarge	0.341 USD	4	0.085 USD

執行個體類型	us-east-1a	執行個體權重	每個單位小時的價格
c5.12xlarge	0.779 USD	12	0.065 USD
c5.18xlarge	1.207 USD	18	0.067 USD
c5.24xlarge	1.555 USD	24	0.065 USD

使用多個啟動範本

除了使用多個執行個體類型以外，您也可以使用多個啟動範本。

例如，假設為運算密集型應用程式設定 Auto Scaling 群組，並且想要混合使用 C5、C5a 和 C6g 執行個體類型。不過，C6g 執行個體具有以 64 位元 Arm 架構為基礎的 AWS Graviton 處理器，而 C5 和 C5a 執行個體在 64 位元 Intel x86 處理器上執行。C5 和 C5a 執行個體的 AMI 均適用於這兩種執行個體中的任一種，但不適用於 C6g 執行個體。若要解決此問題，請對 C6g 執行個體使用不同的啟動範本。您仍然可以對 C5 和 C5a 執行個體使用相同的啟動範本。

本節包含使用 AWS CLI 執行與使用多個啟動範本相關任務的程序。目前，僅在使用 AWS CLI 或開發套件時才能使用此功能，無法透過主控台使用。

目錄

- [設定 Auto Scaling 群組以使用多個啟動範本](#)
- [相關資源](#)

設定 Auto Scaling 群組以使用多個啟動範本

您可以設定 Auto Scaling 群組以使用多個啟動範本，如下列範例所示。

設定新的 Auto Scaling 群組以使用多個啟動範本 (AWS CLI)

使用 [create-auto-scaling-group](#) 命令。例如，以下命令會建立新的 Auto Scaling 群組。其指定了 c5.large、c5a.large 以及 c6g.large 執行個體類型，並為 c6g.large 執行個體類型定義新的啟動範本，確保使用適當的 AMI 來啟動 Arm 執行個體。Amazon EC2 Auto Scaling 群組在滿足隨需容量時，會使用執行個體類型順序決定先使用哪一種執行個體類型。

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


config.json 檔案包含下列內容。

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template-for-x86",
        "Version": "$Latest"
      },
    },
    "Overrides": [
      {
        "InstanceType": "c6g.large",
        "LaunchTemplateSpecification": {
          "LaunchTemplateName": "my-launch-template-for-arm",
          "Version": "$Latest"
        }
      },
      {
        "InstanceType": "c5.large"
      },
      {
        "InstanceType": "c5a.large"
      }
    ]
  },
  "InstancesDistribution": {
    "OnDemandBaseCapacity": 1,
    "OnDemandPercentageAboveBaseCapacity": 50,
    "SpotAllocationStrategy": "capacity-optimized"
  },
  "MinSize": 1,
  "MaxSize": 5,
  "DesiredCapacity": 3,
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags": [ ]
}
```

設定現有的 Auto Scaling 群組以使用多個啟動範本 (AWS CLI)

使用 [update-auto-scaling-group](#) 命令。例如，下列命令會將名為 *my-launch-template-for-arm* 的啟動範本指派給名為 *my-asg* 的 Auto Scaling 群組的 *c6g.large* 執行個體類型。

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

config.json 檔案包含下列內容。

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "Overrides": [
        {
          "InstanceType": "c6g.large",
          "LaunchTemplateSpecification": {
            "LaunchTemplateName": "my-launch-template-for-arm",
            "Version": "$Latest"
          }
        },
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        }
      ]
    }
  }
}
```

若要設定新的 Auto Scaling 群組，以使用具有屬性型執行個體類型選擇的多個啟動範本 (AWS CLI)

使用 [create-auto-scaling-group](#) 命令。例如，下列命令會為具有 ARM AMI 的 AWS Graviton 執行個體指定啟動範本，並為具有 x86 AMI 的 AMD 或 Intel 型執行個體指定額外的啟動範本，以建立新的 Auto Scaling 群組。然後，它會使用[屬性型執行個體選取](#)兩次，為每個 CPU 架構選擇各種執行個體類型。您可以使用 update-autoscaling-group 命令，將類似的組態新增至現有的 Auto Scaling 群組。 [update-autoscaling-group](#)

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

config.json 檔案包含下列內容。

```
{
  "AutoScalingGroupName": "my-asg",
```

```

"MixedInstancesPolicy":{
  "LaunchTemplate":{
    "LaunchTemplateSpecification":{
      "LaunchTemplateName":"my-launch-template-for-arm",
      "Version":"$Latest"
    },
    "Overrides":[
      {
        "InstanceRequirements": {
          "VCpuCount": {"Min": 2},
          "MemoryMiB": {"Min": 2048},
          "CpuManufacturers": ["amazon-web-services"]
        }
      },
      {
        "InstanceRequirements": {
          "VCpuCount": {"Min": 2},
          "MemoryMiB": {"Min": 2048},
          "CpuManufacturers": ["intel", "amd"]
        },
        "LaunchTemplateSpecification": {
          "LaunchTemplateName": "my-launch-template-for-x86",
          "Version": "$Latest"
        }
      }
    ]
  },
  "InstancesDistribution":{
    "OnDemandPercentageAboveBaseCapacity": 0,
    "SpotAllocationStrategy": "price-capacity-optimized"
  },
  "MinSize":1,
  "MaxSize":10,
  "DesiredCapacity":6,
  "VPCZoneIdentifier":"subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags":[ ]
}

```

驗證 Auto Scaling 群組的啟動範本

請使用以下其中一個命令：

- [describe-auto-scaling-groups](#) (AWS CLI)

- [Get-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

相關資源

您可以在 [AWS re : Post](#) 上的範本中找到使用屬性型執行個體類型選取來指定多個啟動 CloudFormation 範本的範例。

使用啟動組態建立 Auto Scaling 群組

Important

限制:

- 自 2023 年 1 月 1 日起，啟動組態不再支援新的 Amazon EC2 執行個體類型。這包括支援在初始區域啟動 AWS 區域 後新增至 的任何執行個體類型。
- 在 2023 年 6 月 1 日當天或之後建立的帳戶無法使用 主控台 建立新的啟動組態。
- 2024 年 10 月 1 日當天或之後建立的帳戶無法使用任何方法 (主控台 AWS CLI、API 或 CloudFormation) 建立新的啟動組態。

遷移以啟動範本，以確保您現在或未來不需要建立新的啟動組態。如需將 Auto Scaling 群組遷移至啟動範本的資訊，請參閱 [遷移 Auto Scaling 群組以啟動範本](#)。

如果您已建立啟動組態或 EC2 執行個體，則可以建立將啟動組態用作其 EC2 執行個體的組態範本的 Auto Scaling 群組。啟動組態指定了執行個體的 AMI ID、執行個體類型、金鑰對、安全群組和區塊型裝置映射等資訊。如需建立啟動組態的資訊，請參閱 [建立啟動組態](#)。

您必須擁足夠的權限來建立 Auto Scaling 群組。您還必須擁有足夠的權限來建立 Amazon EC2 Auto Scaling 用於代表您執行動作的服務連結角色 (如果該角色尚不存在)。有關管理員授予您權限時可用作參考的 IAM 政策範例，請參閱 [身分型政策範例](#)。

目錄

- [使用啟動組態建立 Auto Scaling 群組](#)
- [使用 從現有執行個體建立 Auto Scaling 群組 AWS CLI](#)

使用啟動組態建立 Auto Scaling 群組

Important

對於那些尚未從啟動組態遷移至啟動範本的客戶，我們會提供啟動組態的資訊。如需將 Auto Scaling 群組遷移至啟動範本的詳細資訊，請參閱 [遷移 Auto Scaling 群組以啟動範本](#)。

建立 Auto Scaling 群組時，您必須指定必要資訊，以設定 Amazon EC2 執行個體、執行個體的可用區域和 VPC 子網路、所需的容量以及最小和最大容量限制。

下列程序示範如何使用啟動組態建立 Auto Scaling 群組。在啟動組態建立後，您不能對它進行修改，但可以利用 Auto Scaling 群組取代啟動組態。如需詳細資訊，請參閱 [變更 Auto Scaling 群組的啟動組態](#)。

先決條件

- 您必須已經建立啟動組態。如需詳細資訊，請參閱 [建立啟動組態](#)。

若要使用啟動組態建立 Auto Scaling 群組 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在畫面頂端的導覽列上，選擇 AWS 區域 您在建立啟動組態時所使用的相同。
3. 選擇建立 Auto Scaling 群組。
4. 在選擇啟動範本或組態頁面上，針對 Auto Scaling 群組名稱，請輸入 Auto Scaling 群組的名稱。
5. 選擇啟動組態，請執行下列動作：
 - a. 對於 Launch Template (啟動範本)，選擇 Switch to launch configuration (切換至啟動組態)。
 - b. 對於 Launch configuration (啟動組態)，選擇現有的啟動組態。
 - c. 確認您的啟動組態支援您打算使用的所有選項，然後選擇 Next (下一步)。
6. 在 (配置設定) Configure instance launch options (設定執行個體啟動選項) 頁面，Network (網路) 下，對於 VPC，選擇 VPC。Auto Scaling 群組必須與您在啟動組態中指定的安全群組建立在同一個 VPC 中。
7. 對於 (子網路)、Availability Zones and subnets (可用區域和子網路)，請選擇指定 VPC 中的一個或多個子網路。在多個可用區域中，請使用子網路，以獲得高可用性。如需詳細資訊，請參閱 [選擇 VPC 子網路時的注意事項](#)。

8. 選擇下一步。

或者，您可以接受其他的預設值，然後選擇 Skip to review (跳過以進行檢閱)。

9. (選用) 在 Configure advanced options (設定進階選項) 頁面上，設定下列選項，然後選擇 Next (下一步)：

- a. (選用) 對於運作狀態檢查、其他運作狀態檢查類型，選取開啟 Amazon EBS 運作狀態檢查。如需詳細資訊，請參閱[使用運作狀態檢查監控 Amazon EBS 磁碟區受損的 Auto Scaling 執行個體](#)。
- b. (選用) 針對運作狀態檢查寬限期，輸入時長，以秒為單位。此時間量是 Amazon EC2 Auto Scaling 在進入 InService 狀態後檢查執行個體的運作狀態所需的等待時間。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。
- c. 在 Additional settings (其他設定)、Monitoring (監控) 下，選擇是否啟用 CloudWatch 群組指標收集。這些指標提供了可作為潛在問題之指標的度量，例如終止執行個體數量或擱置執行個體的數量。如需詳細資訊，請參閱[監控 Auto Scaling 群組和執行個體的 CloudWatch 指標](#)。
- d. 針對啟用預設執行個體暖機期，選取此選項，然後選擇應用程式的暖機期。如果您正在建立具有擴展政策的 Auto Scaling 群組，則預設執行個體暖機功能會改進用於動態擴展的 Amazon CloudWatch 指標。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

10. (選用) 在 Configure group size and scaling policies (設定群組大小和擴展政策) 頁面上，設定下列選項，然後選擇 Next (下一步)：

- a. 針對所需容量，在群組大小下輸入要啟動之執行個體的初始數量。
- b. 在擴展區段的擴展限制下，如果所需容量的新值大於所需容量下限和所需容量上限，則所需容量上限會自動增加到新的所需容量值。您可視需要變更這些限制。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
- c. 針對 Automatic scaling，選擇是否要建立目標追蹤擴展政策。您也可以在建建立 Auto Scaling 群組後建立此政策。

如果您選擇目標追蹤擴展政策，請依照 [建立目標追蹤擴展政策](#) 中的指示建立政策。

- d. 針對執行個體維護政策，選擇是否要建立執行個體維護政策。您也可以在建建立 Auto Scaling 群組後建立此政策。請遵循 [設定執行個體維護政策](#) 中的指示建立政策。
- e. 在 Instance scale-in protection (執行個體縮減保護) 下，選擇是否啟用執行個體縮減保護。如需詳細資訊，請參閱 [使用執行個體縮減保護來控制執行個體終止](#)。

11. (選用) 若想接收通知，請針對 Add notification (新增通知) 設定通知，然後選擇 Next (下一步)。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 的 Amazon SNS 通知選項 Amazon EC2 Auto Scaling](#)。

12. (選用) 若要新增標籤，選取 Add tag (新增標籤)、提供每個標籤的標籤鍵和標籤值，然後選擇 Next (下一步)。如需詳細資訊，請參閱 [標記 Auto Scaling 群組和執行個體](#)。
13. 在 Review (檢閱) 頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

使用命令列建立 Auto Scaling 群組

您可以使用下列其中一項命令：

- [create-auto-scaling-group](#) (AWS CLI)
- [New-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

使用 從現有執行個體建立 Auto Scaling 群組 AWS CLI

Important

對於那些尚未從啟動組態遷移至啟動範本的客戶，我們會提供啟動組態的資訊。如需將 Auto Scaling 群組遷移至啟動範本的詳細資訊，請參閱 [遷移 Auto Scaling 群組以啟動範本](#)。

如果這是您第一次建立 Auto Scaling 群組，則建議您使用主控台從現有 EC2 執行個體建立啟動範本。然後使用啟動範本來建立新的 Auto Scaling 群組。相關程序請參閱 [使用 Amazon EC2 啟動精靈建立 Auto Scaling 群組](#)。

以下程序說明如何透過指定用作啟動其他執行個體之基礎的現有執行個體來建立 Auto Scaling 群組。建立 EC2 執行個體需要多個參數，例如 Amazon Machine Image (AMI) ID、執行個體類型、金鑰對和安全群組。Amazon EC2 Auto Scaling 也會使用所有這些資訊，在需要擴展時代表您啟動執行個體。此資訊存放在啟動範本或啟動組態中。

使用現有執行個體時，Amazon EC2 Auto Scaling 會建立 Auto Scaling 群組，該群組根據同時建立的啟動組態啟動執行個體。新的啟動組態與 Auto Scaling 群組名稱相同，其中包含已識別執行個體的某些組態詳細資訊。

以下組態詳細資訊會從已識別執行個體複製到啟動組態：

- AMI ID
- 執行個體類型
- 金鑰對
- Security groups (安全群組)

- IP 地址類型 (公有或私有)
- IAM 執行個體設定檔 (如果適用)
- 監控 (true 或 false)
- EBS 最佳化 (true 或 false)
- 租用設定 (若在 VPC 中啟動的話 (共享或專用))
- 核心 ID 和 RAM 磁碟區 ID (如適用)
- 使用者資料 (若指定)
- Spot (最高) 價格

VPC 子網路和可用區域會從已識別的執行個體複製到 Auto Scaling 群組本身的資源定義。

如果該已識別執行個體是位於置放群組，則新 Auto Scaling 群組會將執行個體啟動至與已識別執行個體相同的置放群組。由於啟動組態設定不允許指定置放群組，因此置放群組會複製到新 Auto Scaling 群組的 PlacementGroup 屬性。

以下組態詳細資訊不是從已識別執行個體複製的。

- 儲存：區塊型儲存設備 (EBS 磁碟區和執行個體存放區磁碟區) 不會從已識別執行個體中複製。相對地，作為建立 AMI 的一部分而建立的區塊型設備映射會決定使用哪些設備。
- 網路介面的數量：網路介面不會從已識別執行個體中複製。相對地，Amazon EC2 Auto Scaling 會使用其預設設定建立一個網路介面，即主要網路介面 (eth0)。
- 執行個體中繼資料選項：不會從已識別執行個體複製可存取的中繼資料、中繼資料版本和字符回應躍點限制設定。相對地，Amazon EC2 Auto Scaling 會使用其預設設定。如需詳細資訊，請參閱[設定執行個體中繼資料選項](#)。
- 負載平衡器：如果已識別執行個體註冊一或多個負載平衡器，則負載平衡器的相關資訊不會複製到新 Auto Scaling 群組的負載平衡器或目標群組屬性。
- 標籤：如果已識別執行個體具有標籤，則此標籤並不會複製到新 Auto Scaling 群組的 Tags 屬性。

先決條件

EC2 執行個體必須符合下列標準：

- 該執行個體不屬於另一個 Auto Scaling 群組。
- 執行個體的狀態為 running。
- 用於啟動執行個體的 AMI 必須仍然存在。

從 EC2 執行個體 (AWS CLI) 建立 Auto Scaling 群組

下列程序說明如何使用 CLI 命令從 EC2 執行個體建立 Auto Scaling 群組。

此程序不會將執行個體新增到 Auto Scaling 群組。對於要連接的執行個體，您必須在建立 Auto Scaling 群組之後執行 [attach-instances](#) 命令。

在開始前，請先使用 Amazon EC2 主控台或 [describe-instances](#) 命令找到 EC2 執行個體的 ID。

使用您目前的執行個體做為範本

- 使用以下 [create-auto-scaling-group](#) 命令，從該 EC2 執行個體 `i-123456789abcdefg0` 建立 Auto Scaling 群組 `my-asg-from-instance`。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-from-instance \
  --instance-id i-123456789abcdefg0 --min-size 1 --max-size 2 --desired-capacity 2
```

若要驗證已啟動新執行個體的 Auto Scaling 群組

- 使用下列 [describe-auto-scaling-groups](#) 命令來確認 Auto Scaling 群組已成功建立。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg-from-instance
```

以下範例回應顯示所需的群組容量為 2、群組擁有 2 個執行中的執行個體；另外，啟動組態的名稱為 `my-asg-from-instance`。

```
{
  "AutoScalingGroups":[
    {
      "AutoScalingGroupName":"my-asg-from-instance",
      "AutoScalingGroupARN":"arn",
      "LaunchConfigurationName":"my-asg-from-instance",
      "MinSize":1,
      "MaxSize":2,
      "DesiredCapacity":2,
      "DefaultCooldown":300,
      "AvailabilityZones":[
        "us-west-2a"
      ],
    }
  ],
}
```

```

    "LoadBalancerNames": [],
    "TargetGroupARNs": [],
    "HealthCheckType": "EC2",
    "HealthCheckGracePeriod": 0,
    "Instances": [
      {
        "InstanceId": "i-34567890abcdef012",
        "InstanceType": "t2.micro",
        "AvailabilityZone": "us-west-2a",
        "LifecycleState": "InService",
        "HealthStatus": "Healthy",
        "LaunchConfigurationName": "my-asg-from-instance",
        "ProtectedFromScaleIn": false
      },
      {
        "InstanceId": "i-012345abcdefg6789",
        "InstanceType": "t2.micro",
        "AvailabilityZone": "us-west-2a",
        "LifecycleState": "InService",
        "HealthStatus": "Healthy",
        "LaunchConfigurationName": "my-asg-from-instance",
        "ProtectedFromScaleIn": false
      }
    ],
    "CreatedTime": "2020-10-28T02:39:22.152Z",
    "SuspendedProcesses": [ ],
    "VPCZoneIdentifier": "subnet-0abc1234",
    "EnabledMetrics": [ ],
    "Tags": [ ],
    "TerminationPolicies": [
      "Default"
    ],
    "NewInstancesProtectedFromScaleIn": false,
    "ServiceLinkedRoleARN": "arn",
    "TrafficSources": [ ]
  }
]
}

```

若要檢視啟動組態

- 使用下列 [describe-launch-configurations](#) 命令來檢視啟動組態的詳細資訊。

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

下列為範例輸出：

```
{
  "LaunchConfigurations":[
    {
      "LaunchConfigurationName":"my-asg-from-instance",
      "LaunchConfigurationARN":"arn",
      "ImageId":"ami-234567890abcdefgh",
      "KeyName":"my-key-pair-uswest2",
      "SecurityGroups":[
        "sg-12abcdefgh3456789"
      ],
      "ClassicLinkVPCSecurityGroups":[ ],
      "UserData":"",
      "InstanceType":"t2.micro",
      "KernelId":"",
      "RamdiskId":"",
      "BlockDeviceMappings":[ ],
      "InstanceMonitoring":{
        "Enabled":true
      },
      "CreatedTime":"2020-10-28T02:39:22.321Z",
      "EbsOptimized":false,
      "AssociatePublicIpAddress":true
    }
  ]
}
```

若要終止執行個體

- 若您不再需要執行個體，可將它終止。下列 [terminate-instances](#) 命令可終止執行個體 `i-123456789abcdefgh0`。

```
aws ec2 terminate-instances --instance-ids i-123456789abcdefgh0
```

Amazon EC2 執行個體終止後，就無法再重新啟動該執行個體。終止之後，它的資料會消失，磁碟區也無法連接至任何執行個體。若要進一步了解終止執行個體，請參閱《Amazon EC2 使用者指南》中的[終止執行個體](#)。

同步啟動執行個體

Amazon EC2 Auto Scaling 提供兩種在 Auto Scaling 群組中啟動執行個體的方法：使用 LaunchInstances API 的非同步擴展行為和同步佈建。

透過同步佈建，您可以使用 LaunchInstances API 來請求特定可用區域中的特定執行個體數量。同步佈建提供下列優點：

- 針對特定可用區域中容量可用性的立即意見回饋
- 精確控制哪些可用區域執行個體在 中啟動
- 可立即在協同運作系統中使用的確定性執行個體 IDs
- 根據實際容量限制進行即時擴展決策
- 透過消除非同步 Auto Scaling 啟動的等待時間，加快擴展速度

使用非同步 Auto Scaling 時，當您變更所需的容量或擴展政策觸發時，Amazon EC2 Auto Scaling 會處理擴展請求並在背景啟動執行個體。您必須監控擴展活動或描述 Auto Scaling 群組，以判斷執行個體何時成功啟動。

Note

- LaunchInstances API 僅適用於使用啟動範本的 Auto Scaling 群組。不支援使用啟動組態的 Auto Scaling 群組。如果您的 Auto Scaling 群組使用啟動組態，您必須在使用同步佈建之前遷移至啟動範本。
- LaunchInstances API 僅支援具有完全隨需或完全 Spot 購買選項的混合執行個體政策。不支援結合隨需執行個體和 Spot 執行個體的混合政策。
- 對於涵蓋多個可用區域的 Auto Scaling 群組，您必須指定目標可用區域或子網路。對於單一可用區群組，此參數是選用的。

同步佈建和非同步擴展

同步佈建

當您使用 LaunchInstances API 時，Amazon EC2 Auto Scaling：

- 立即嘗試使用 CreateFleet 啟動請求的執行個體
- 等待 CreateFleet 傳回執行個體 IDs 然後再回應
- 傳回執行個體 IDs、執行個體類型和可用區域的成功資訊
- 傳回特定錯誤代碼和失敗的詳細資訊
- 提供即時意見回饋，實現即時擴展決策

非同步擴展

當您使用非同步 Auto Scaling 方法，例如變更所需的容量或使用擴展政策時，Amazon EC2 Auto Scaling：

- 在 API 中更新所需的容量，但不會立即傳回執行個體
- Plans 執行個體會自動跨可用區域啟動
- 透過背景工作流程啟動執行個體
- 自動分配容量到多個可用區域以平衡
- 使用內建重試邏輯處理啟動失敗

您必須輪詢擴展活動或描述 Auto Scaling 群組，以檢查啟動操作的狀態。

限制及考量

使用同步佈建時，請記住下列注意事項和限制：

- 啟動後的執行個體狀態 – API 傳回的執行個體處於擱置狀態。它們在後續工作流程或生命週期關聯期間仍可能會失敗。成功的 API 回應表示 EC2 已接受啟動請求並傳回執行個體 IDs。執行個體不會自動視為完全準備好處理工作負載，而且必須完成標準 EC2 和 Auto Scaling 生命週期程序。
- 暖集區限制 – 目前不支援具有暖集區的 Auto Scaling 群組。如果您嘗試在已設定暖集區的 Auto Scaling 群組上呼叫 LaunchInstances API，API 會執行冷啟動，而不是使用暖集區執行個體，並傳回 `UnsupportedOperation` 錯誤。如需冷啟動的詳細資訊，請參閱[暖集區的限制](#)。

- **API 逾時和重試** – 如果基礎 CreateFleet 操作所花費的時間超過預期，API 可能會逾時並傳回冪等字符。您可以使用相同的 ClientToken 來追蹤原始啟動操作，或使用 describe-instances 搭配用戶端字符來檢查啟動的執行個體。
- **可用區域限制** – 如果您的 Auto Scaling 群組跨越多個可用區域，且已啟用可用區域重新平衡，則同步啟動執行個體可能會導致操作衝突：
 - **每次呼叫的單一可用區限制** – 每個 LaunchInstances API 呼叫只能以一個可用區域為目標，即使您的 Auto Scaling 群組跨越多個區域。
 - **AZ 重新平衡衝突** - 如果您的 Auto Scaling 群組已啟用 AZ 重新平衡，跨不同 AZs 循序呼叫可能會觸發其他非同步啟動，導致執行個體超過預期。請考慮暫停 AZ 重新平衡，以精確控制容量。如需詳細資訊，請參閱 [暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。
- **部分成功案例** – 如果只有部分請求的容量可用，則 LaunchInstances API 可能會傳回部分成功，這是正常的 EC2 行為。API 會傳回成功啟動的執行個體，以及失敗啟動的錯誤詳細資訊。對於需要所有執行個體一起啟動的使用案例（例如需要相同 AZ 中所有執行個體以實現低延遲的應用程式），您將需要終止部分啟動的執行個體，並在不同的 AZ 中重試。為容量敏感工作負載設計重試邏輯時，請考慮此行為。
- **執行個體權重** – 如果您的 Auto Scaling 群組使用執行個體權重，RequestedCapacity 參數代表加權容量單位，而不是執行個體的數量。實際啟動的執行個體數量取決於選取的執行個體類型及其設定的權重。無論請求的加權容量為何，EC2 Auto Scaling 會限制每個 API 呼叫啟動至 100 個執行個體。
- **混合執行個體類型** – LaunchInstances API 會使用 Auto Scaling 群組的現有混合執行個體政策來決定要啟動的執行個體類型。API 會根據您群組的配置策略和執行個體類型優先順序來啟動執行個體。

使用同步佈建啟動執行個體

您可以使用 LaunchInstances API，在 Auto Scaling 群組中同步啟動特定數量的執行個體。API 會在您指定的可用區域或子網路中啟動執行個體，並立即傳回執行個體 IDs 或錯誤資訊。

先決條件

您必須先具有下列項目，才能使用 LaunchInstances API：

- 使用啟動範本的 Auto Scaling 群組（不支援啟動組態）
- 您必須具有下列 IAM 動作的許可：
 - `autoscaling:LaunchInstances`
 - `ec2:CreateFleet`
 - `ec2:DescribeLaunchTemplateVersions`

使用同步佈建啟動執行個體

您可以透過 啟動具有同步佈建的執行個體 AWS CLI。

AWS CLI

若要使用同步佈建啟動執行個體：

```
aws autoscaling launch-instances \  
    --auto-scaling-group-name group-name \  
    --requested-capacity number \  
    [--availability-zones zone-name] \  
    [--subnet-ids subnet-id] \  
    [--availability-zone-ids zone-id] \  
    [--retry-strategy none|retry-with-group-configuration] \  
    [--client-token token]
```

範例

在特定可用區域中啟動執行個體

```
aws autoscaling launch-instances \  
    --auto-scaling-group-name my-asg \  
    --requested-capacity 3 \  
    --availability-zones us-east-1a \  
    --retry-strategy retry-with-group-configuration
```

在特定子網路中啟動執行個體

```
aws autoscaling launch-instances \  
    --auto-scaling-group-name my-asg \  
    --requested-capacity 2 \  
    --subnet-ids subnet-12345678 \  
    --retry-strategy none \  
    --client-token my-unique-token-123
```

處理回應

成功回應的範例：

```
{
```

```

"AutoScalingGroupName": "my-asg",
"ClientToken": "my-unique-token-123",
"Instances": [
  {
    "InstanceType": "m5.xlarge",
    "AvailabilityZone": "us-east-1a",
    "AvailabilityZoneId": "use1-az1",
    "SubnetId": "subnet-12345678",
    "MarketType": "OnDemand",
    "InstanceIds": ["i-0123456789abcdef0", "i-0fedcba9876543210"]
  }
],
"Errors": []
}

```

出現錯誤的回應範例

```

{
  "AutoScalingGroupName": "my-asg",
  "ClientToken": "my-unique-token-123",
  "Instances": [],
  "Errors": [
    {
      "InstanceType": "m5.large",
      "AvailabilityZone": "us-east-1a",
      "AvailabilityZoneId": "use1-az1",
      "SubnetId": "subnet-12345678",
      "MarketType": "OnDemand",
      "ErrorCode": "InsufficientInstanceCapacity",
      "ErrorMessage": "There is not enough capacity to fulfill your request for instance type 'm5.large' in 'us-east-1a'"
    }
  ]
}

```

處理啟動失敗和重試

當 LaunchInstances API 遇到失敗時，您可以使用冪等字符和適當的重試政策來實作重試策略。

您可以使用用戶端字符參數重試請求。您也可以使用下列重試策略：

- **RetryStrategy: none** (預設) - 如果 API 呼叫失敗，Auto Scaling 群組所需的容量保持不變，不會自動重試。
- **RetryStrategy: retry-with-group-configuration** - 如果 API 呼叫失敗，Auto Scaling 群組所需的容量會增加請求的數量，Auto Scaling 將使用群組的標準組態和程序自動重試啟動執行個體。

的重試行為 **RetryStrategy: retry-with-group-configuration** 取決於失敗類型：

- **驗證錯誤**：由於無法繼續操作，因此不會增加所需的容量。例如，無效的參數或不支援的組態。
- **容量錯誤**：所需容量增加，Auto Scaling 將使用群組的正常擴展程序，以非同步方式重試啟動執行個體。

將用戶端字符用於冪等性

`client-token` 參數可確保冪等操作，並啟用啟動請求的安全重試。

關鍵行為：

- 用戶端字符在初始請求的生命週期為 8 小時
- 在 8 小時內使用相同的用戶端字符重試會傳回快取的回應，而不是啟動新的執行個體
- 8 小時後，相同的用戶端字符將啟動新的啟動操作

更新 Auto Scaling 群組

您可以更新 Auto Scaling 群組的大部分詳細資料。您無法更新 Auto Scaling 群組的名稱或變更其名稱 AWS 區域。

更新 Auto Scaling 群組 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選擇 Auto Scaling 群組以顯示該群組的相關資訊，其中包含詳細資料、活動、自動擴展、執行個體管理、監控和執行個體重新整理等索引標籤。
3. 選擇您感興趣的組態區域的索引標籤，並視需要更新設定。針對您編輯的每個設定，選擇更新以儲存對 Auto Scaling 群組組態的變更。
 - 詳細資訊索引標籤

這些為 Auto Scaling 群組的一般設定。您可以使用與建立 Auto Scaling 群組期間相同的方式來編輯和管理這些設定。

進階組態區段中有一些在建立群組時無法使用的選項，例如[終止政策](#)、[冷卻時間](#)、[暫停的處理程序](#)和[執行個體生命週期上限](#)。您也可以檢視但不能編輯 Auto Scaling 群組的置放群組和[服務連結角色](#)。

- 整合索引標籤

- 負載平衡 – [Elastic Load Balancing](#)

如果群組與 Elastic Load Balancing 資源關聯，在變更可用區域之前請參閱 [新增可用區域](#)。負載平衡器的某些限制可能會阻止您將群組可用區域的變更套用至負載平衡器的可用區域。

- VPC Lattice 整合選項 – [VPC Lattice](#)

- ARC 區域轉移 – [Auto Scaling 群組區域轉移](#)

- 自動擴展索引標籤

- 動態擴展政策 – [動態擴展政策](#)

- 預測擴展政策 – [預測擴展政策](#)

- 排程動作 – [排程動作](#)

- 執行個體管理索引標籤

- 生命週期掛鉤 – [生命週期掛鉤](#)

- 暖集區 – [暖集區](#)

- 活動索引標籤

- 活動通知 – [Amazon SNS 通知](#)

- 監控索引標籤

- 此索引標籤中只有一個選項，可讓您啟用或停用 [CloudWatch 群組指標收集](#)。

使用命令列更新 Auto Scaling 群組

您可以使用下列其中一項命令：

- [update-auto-scaling-group](#) (AWS CLI)
- [Update-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

更新 Auto Scaling 執行個體

如果您將新的啟動範本或啟動組態與 Auto Scaling 群組進行關聯，所有新的執行個體都會收到更新後的組態。現有的執行個體會採用原始的啟動組態繼續運作。要將您的變更套用至現有的執行個體，您有下列選項：

- 啟動執行個體重新整理來取代較舊的執行個體。如需詳細資訊，請參閱[使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。
- 等待擴展活動根據[終止政策](#)逐步將較舊的執行個體取代為較新的執行個體。
- 手動終止它們，以使用您的 Auto Scaling 群組取代它們。

Note

您可以將下列執行個體屬性指定為啟動範本或啟動組態的一部分，從而變更這些執行個體屬性：

- Amazon Machine Image (AMI)
- 區塊型儲存裝置
- 金鑰對
- 執行個體類型
- 安全群組
- 使用者資料
- 監控
- IAM 執行個體描述檔
- 安置租用
- kernel
- 記憶體虛擬磁碟
- 執行個體是否具備公有 IP 地址
- 可用區域分佈策略

Auto Scaling 群組配置策略和容量變更

當您變更 Auto Scaling 群組配置策略時，不會取代現有的執行個體。因橫向擴展事件而啟動的任何新執行個體都會遵循新的配置策略。任何未來的擴展事件都將遵循[終止政策](#)，並在終止政策設定為 Default 或 時使用新的配置策略 AllocationStrategy。例如，如果您將配置策略從 變更為 lowest-price price-capacity-optimized，則可能沒有任何執行個體終止，但任何新的執行個體都會使用新的配置策略啟動。執行個體類型變更不會影響現有的執行個體。

當您變更 [OnDemandBaseCapacity](#) 或 [OnDemandPercentageAboveBaseCapacity](#) 等特定參數時，如果隨需執行個體和 Spot 執行個體的百分比不符合新規格，Auto Scaling 會自動重新平衡。例如，假設 Auto Scaling 群組的 OnDemandPercentageAboveBaseCapacity 設定為 50% 隨需執行個體和 50% Spot 執行個體。然後，OnDemandPercentageAboveBaseCapacity 會提高到 100% 隨需執行個體。Auto Scaling 群組會透過啟動新的隨需執行個體並終止 Spot 執行個體來主動重新平衡。您定義的[執行個體維護政策](#)會決定啟動和終止活動的順序。

標記 Auto Scaling 群組和執行個體

標籤是您指派或 AWS 指派給 AWS 資源的自訂屬性標籤。每個 標籤都有兩個部分：

- 標籤鍵 (例如 costcenter、environment 或 project)。
- 選用欄位，稱為標籤值 (例如 111122223333 或 production)。

標籤可協助您執行以下操作：

- 追蹤您的 AWS 成本。您可以在 AWS 帳單與成本管理 儀表板上啟用這些標籤。AWS 使用標籤來分類您的成本，並為您提供每月成本分配報告。如需詳細資訊，請參閱《AWS Billing 使用者指南》中的[使用成本分配標籤](#)。
- 根據標籤控制對 Auto Scaling 群組的存取。您可以在 IAM 政策中使用條件，根據該群組的標籤來控制對 Auto Scaling 群組的存取。如需詳細資訊，請參閱[安全標籤](#)。
- 根據您所新增的標籤篩選和搜索 Auto Scaling 群組。如需詳細資訊，請參閱[使用標籤篩選 Auto Scaling 群組](#)。
- 識別和組織您的 AWS 資源。許多 AWS 服務 支援標記，因此您可以將相同的標籤指派給來自不同服務的資源，以指出資源相關。

您可以標記新的或現有 Auto Scaling 群組。您也可以從 Auto Scaling 群組傳播標籤到該群組啟動的 EC2 執行個體。

標籤不會傳播到 Amazon EBS 磁碟區。若要新增標籤至 Amazon EBS 磁碟區，請在啟動範本中指定標籤。如需詳細資訊，請參閱[為 Auto Scaling 群組建立啟動範本](#)。

您可以透過 AWS 管理主控台、AWS CLI 或 SDKs 建立和管理標籤。

目錄

- [標籤命名和使用限制](#)
- [EC2 執行個體標記生命週期](#)
- [標記 Auto Scaling 群組](#)
- [刪除標籤](#)
- [安全標籤](#)
- [控制對標籤的存取](#)
- [使用標籤篩選 Auto Scaling 群組](#)

標籤命名和使用限制

以下基本限制適用於 標籤：

- 每一資源標籤數最多為 50。
- 單一呼叫中，您所能新增或移除最多的標籤數量為 25。
- 鍵長度上限為 128 個 Unicode 字元。
- 值長度上限為 256 個 Unicode 字元。
- 標籤鍵與值皆區分大小寫。作為最佳實務，請決定大寫標籤的策略，並一致地在所有資源類型中實作該策略。
- 請勿在標籤名稱或值中使用 `aws:` 字首，因為它已保留供 AWS 使用。您不能編輯和刪除具有此字首的標籤名稱或值，它們不會計算在每個資源配額的標籤之內。

EC2 執行個體標記生命週期

如果您已選用將標籤傳播到您的 EC2 執行個體，標籤會依以下方式管理：

- 當 Auto Scaling 群組啟動執行個體時，其會在建立資源期間 (而非在建立資源後) 對執行個體新增標籤。
- Auto Scaling 群組會自動將標籤新增至執行個體，且該執行個體具有 `aws:autoscaling:groupName` 鍵和 Auto Scaling 群組名稱值。

- 如果在啟動範本中指定執行個體標籤，並選擇將群組的標籤傳播到其執行個體，則會合併所有標籤。如果為啟動範本中的標籤和 Auto Scaling 群組中的標籤指定了相同的標籤鍵，則以群組中的標籤值為優先。
- 在連接現有執行個體時，Auto Scaling 群組會為執行個體新增標籤，以相同標籤鍵覆寫現有執行個體的標籤。它還會新增具有 `aws:autoscaling:groupName` 鍵和 Auto Scaling 群組名稱值的標籤。
- 從 Auto Scaling 群組分離執行個體時，它只會移除 `aws:autoscaling:groupName` 標籤。

標記 Auto Scaling 群組

在將標籤新增到 Auto Scaling 群組時，您可以選擇是否要新增已啟動的執行個體到 Auto Scaling 群組。假設您修改了標籤，在變更後完成後，標籤的更新版本會新增到 Auto Scaling 群組中已啟動的執行個體。如果您建立或修改 Auto Scaling 群組的標籤，這些變更不會套用到已在 Auto Scaling 群組中執行的執行個體。

目錄

- [新增或修改標籤 \(主控台\)](#)
- [新增或修改標籤 \(AWS CLI\)](#)

新增或修改標籤 (主控台)

若要在建立時標記 Auto Scaling 群組

在使用 Amazon EC2 主控台建立 Auto Scaling 群組時，您可以在 Create Auto Scaling Group (建立 Auto Scaling 群組) 精靈中的 Add tags (新增標籤) 頁面上指定標籤鍵和值。若要將標籤傳播至 Auto Scaling 群組中啟動的執行個體，請確定您有保留選取該標籤的 Tag new instances (標記新的執行個體) 選項。否則，您可加以取消選取。

在現有 Auto Scaling 群組中新增或修改標籤

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，依次選擇 Tags (標籤)、Edit (編輯)。
4. 若要修改現有標籤，請編輯 Key (鍵) 和 Value (值)。

- 若要新增標籤，請選擇 Add tag (新增標籤)，然後編輯 Key (鍵) 和 Value (值)。您可以繼續選取 Tag new instances (標記新的執行個體)，以自動將標籤新增到 Auto Scaling 群組中啟動的執行個體。否則，請取消該選取。
- 當您完成新增標籤的作業時，請選擇 Update (更新)。

新增或修改標籤 (AWS CLI)

下列範例示範如何在建立 Auto Scaling 群組時使用 AWS CLI 新增標籤，以及新增或修改現有 Auto Scaling 群組的標籤。

若要在建立時標記 Auto Scaling 群組

使用 [create-auto-scaling-group](#) 命令來建立新的 Auto Scaling 群組並新增標籤，例如新增 **environment=production** 至 Auto Scaling 群組。此標籤也會新增至 Auto Scaling 群組中啟動的任何執行個體。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
  --launch-configuration-name my-launch-config --min-size 1 --max-size 3 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
  --tags Key=environment,Value=production,PropagateAtLaunch=true
```

在現有 Auto Scaling 群組中建立或修改標籤

使用 [create-or-update-tags](#) 命令來建立或修改標籤。例如，以下命令會新增 **Name=my-asg** 和 **costcenter=cc123** 標籤。在此變更之後，這兩個標籤也會新增至 Auto Scaling 群組中啟動的任何執行個體。如果具有任一鍵的標籤已存在，則現有的標籤會被取代。Amazon EC2 主控台會將每個執行個體的顯示名稱關聯至為 Name 鍵指定的名稱 (區分大小寫)。

```
aws autoscaling create-or-update-tags \  
  --tags ResourceId=my-asg,ResourceType=auto-scaling-group,Key=Name,Value=my-  
asg,PropagateAtLaunch=true \  
  ResourceId=my-asg,ResourceType=auto-scaling-  
group,Key=costcenter,Value=cc123,PropagateAtLaunch=true
```

描述 Auto Scaling 群組 (AWS CLI) 的標籤

如果您要檢視已套用至特定 Auto Scaling 群組的標籤，可使用以下其中一個 API 命令：

- [describe-tags](#) – 您可以提供 Auto Scaling 群組名稱，以檢視指定群組的標籤清單。

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

以下是回應範例。

```
{
  "Tags": [
    {
      "ResourceType": "auto-scaling-group",
      "ResourceId": "my-asg",
      "PropagateAtLaunch": true,
      "Value": "production",
      "Key": "environment"
    }
  ]
}
```

- [describe-auto-scaling-groups](#) – 您可以提供 Auto Scaling 群組名稱，以檢視指定群組的屬性，包括任何標籤。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下是回應範例。

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn",
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
      "MinSize": 1,
      "MaxSize": 5,
      "DesiredCapacity": 1,
      "...",
      "Tags": [
        {
```



```
        "ResourceType": "auto-scaling-group",
        "ResourceId": "my-asg",
        "PropagateAtLaunch": true,
        "Value": "production",
        "Key": "environment"
    },
    ...
]
```

刪除標籤

您隨時都可以刪除與 Auto Scaling 群組有關聯的標籤。

目錄

- [刪除標籤 \(主控台\)](#)
- [刪除標籤 \(AWS CLI\)](#)

刪除標籤 (主控台)

若要刪除標籤

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，依次選擇 Tags (標籤)、Edit (編輯)。
4. 選擇標籤旁的 Remove (移除)。
5. 選擇更新。

刪除標籤 (AWS CLI)

使用 [delete-tags](#) 命令來刪除標籤。例如，以下命令會刪除包含 **environment** 鍵的標籤。

```
aws autoscaling delete-tags --tags "ResourceId=my-asg,ResourceType=auto-scaling-group,Key=environment"
```

您必須指定標籤鍵，但不需要指定值。如果您指定的值不正確，則此不會刪除此標籤。

安全標籤

使用標籤來確認請求者 (例如 IAM 使用者或角色) 具有建立、修改或刪除特定 Auto Scaling 群組的許可。使用下列一個或多個條件金鑰，在 IAM 政策的條件元素中提供標籤資訊：

- 使用 `autoscaling:ResourceTag/tag-key: tag-value` 允許 (或拒絕) 在有特定標籤的 Auto Scaling 群組上的使用者動作。
- 使用 `aws:RequestTag/tag-key: tag-value` 要求在請求中出現 (或不出現) 特定標籤。
- 使用 `aws:TagKeys [tag-key, ...]` 要求在請求中出現 (或不出現) 特定標籤鍵。

例如，您可以拒絕所有 Auto Scaling 群組的存取，該群組包含鍵 **environment** 和值 **production**，如下列範例所示。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:DeleteAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {"autoscaling:ResourceTag/environment":
production}
      }
    }
  ]
}
```

如需使用條件金鑰來控制存取 Auto Scaling 群組的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 與 IAM 搭配運作的方式](#)。

控制對標籤的存取

使用標籤來確認請求者 (例如 IAM 使用者或角色) 具有新增、修改或刪除 Auto Scaling 群組之標籤的許可。

以下範例 IAM 政策授予主體僅從 Auto Scaling 群組中刪除帶有 **temporary** 金鑰的標籤的權限。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "autoscaling:DeleteTags",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": { "aws:TagKeys": ["temporary] }
      }
    }
  ]
}
```

如需針對 Auto Scaling 群組指定的標籤強制執行限制的 IAM 政策的更多範例，請參閱 [控制可以使用哪些標籤鍵和標籤值](#)。

Note

即使政策可以限制使用者對 Auto Scaling 群組執行標記 (或取消標記) 操作，這也無法阻止在執行個體啟動後對其上面的標籤進行手動變更。如需控制 EC2 執行個體上標籤存取的範例，請參閱《Amazon EC2 使用者指南》中的 [範例：標記資源](#)。

使用標籤篩選 Auto Scaling 群組

以下範例會顯示如何使用具有 [describe-auto-scaling-groups](#) 命令的篩選條件，來描述具有特定標籤的 Auto Scaling 群組。依標籤篩選僅限於 AWS CLI 或 SDK，而且無法從 主控台 使用。

篩選考量

- 您可以在單一請求中指定多個篩選條件和多個篩選條件值。
- 您無法使用萬用字元搭配篩選條件值。
- 篩選條件值區分大小寫。

範例：描述具有特定標籤鍵值對的 Auto Scaling 群組

以下命令顯示如何篩選結果，以便僅顯示具有 **environment=production** 鍵值對的 Auto Scaling 群組。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-value,Values=production
```

以下是回應範例。

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn",
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
      "MinSize": 1,
      "MaxSize": 5,
      "DesiredCapacity": 1,
      "...",
      "Tags": [
        {
          "ResourceType": "auto-scaling-group",
          "ResourceId": "my-asg",
          "PropagateAtLaunch": true,
```

```

        "Value": "production",
        "Key": "environment"
    }
],
...
},
... additional groups ...
]
}

```

或者，您也可以使用 `tag:<key>` 篩選條件指定標籤。例如，以下命令顯示如何篩選結果，以便僅顯示具有 **environment=production** 鍵值對的 Auto Scaling 群組。此篩選條件的格式如下所示：Name=tag:<key>,Values=<value>，使用 <key> 和 <value> 表示標籤鍵值對。

```

aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production

```

您也可以使用 `--query` 選項篩選 AWS CLI 輸出。下列範例顯示如何將先前命令的 AWS CLI 輸出限制為僅群組名稱、最小大小、最大大小和所需的容量屬性。

```

aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production \
  --query "AutoScalingGroups[].{AutoScalingGroupName: AutoScalingGroupName, MinSize: MinSize, MaxSize: MaxSize, DesiredCapacity: DesiredCapacity}"

```

以下是回應範例。

```

[
  {
    "AutoScalingGroupName": "my-asg",
    "MinSize": 0,
    "MaxSize": 10,
    "DesiredCapacity": 1
  },
  ... additional groups ...
]

```

如需篩選的詳細資訊，請參閱AWS Command Line Interface 《使用者指南》中的[篩選 AWS CLI 輸出](#)。

範例：描述具有與指定標籤鍵相符之標籤的 Auto Scaling 群組

以下命令顯示如何篩選結果，以僅顯示具有 **environment** 標籤的 Auto Scaling 群組，而不考慮標籤值。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment
```

範例：描述具有與指定標籤鍵集相符之標籤的 Auto Scaling 群組

以下命令顯示如何篩選結果，以僅顯示具有 **environment** 和 **project** 標籤的 Auto Scaling 群組，而不考慮標籤值。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-key,Values=project
```

範例：描述具有至少一個與指定標籤鍵相符之標籤的 Auto Scaling 群組

以下命令顯示如何篩選結果，以僅顯示具有 **environment** 或 **project** 標籤的 Auto Scaling 群組，而不考慮標籤值。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment,project
```

範例：描述具有指定標籤值的 Auto Scaling 群組

以下命令顯示如何篩選結果，以僅顯示具有 **production** 標籤值的 Auto Scaling 群組，而不考慮標籤鍵。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production
```

範例：描述具有指定標籤值集的 Auto Scaling 群組

以下命令顯示如何篩選結果，以僅顯示具有標籤值 **production** 和 **development** 的 Auto Scaling 群組，而不考慮標籤鍵。

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

```
--filters Name=tag-value,Values=production Name=tag-value,Values=development
```

範例：描述具有至少一個與指定標籤值相符之標籤的 Auto Scaling 群組

以下命令顯示如何篩選結果，以僅顯示具有標籤值 **production** 或 **development** 的 Auto Scaling 群組，而不考慮標籤鍵。

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag-value,Values=production,development
```

範例：描述具有與多個標籤鍵和值相符之標籤的 Auto Scaling 群組

您還可以組合篩選條件來建立自訂 AND 和 OR 邏輯，以執行更複雜的篩選。

以下命令顯示如何篩選結果，以僅顯示具有指定標籤集的 Auto Scaling 群組。一個標籤鍵是 **environment** AND 標籤值是 (**production** OR **development**) AND 另一個標籤鍵是 **costcenter** AND 標籤值是 **cc123**。

```
aws autoscaling describe-auto-scaling-groups \  
  --filters Name=tag:environment,Values=production,development \  
    Name=tag:costcenter,Values=cc123
```

執行個體維護政策

您可以為 Auto Scaling 群組設定執行個體維護政策，以滿足導致執行個體被取代的事件 (例如執行個體重新整理或運作狀態檢查程序) 發生期間的特定容量需求。

例如，假設您擁有一個具有少量執行個體的 Auto Scaling 群組。當運作狀態檢查指示執行個體受損時，您想要避免因終止然後取代執行個體而造成的潛在中斷。使用執行個體維護政策，您便可以確保 Amazon EC2 Auto Scaling 先啟動新的執行個體，然後等待該執行個體完全準備就緒後，再終止運作狀態不佳的執行個體。

執行個體維護政策也可協助您在同一時間取代多個執行個體的情況下，最大限度地減少潛在的中斷。您可以為政策設定狀態良好最低百分比和狀態良好最高百分比參數，而 Auto Scaling 群組只能在取代執行個體時，在該最低至最高範圍內增減容量。範圍越大，可同時取代的執行個體數量便會越多。

目錄

- [Auto Scaling 群組的執行個體維護政策](#)
- [在 Auto Scaling 群組上設定執行個體維護政策](#)

Auto Scaling 群組的執行個體維護政策

本主題提供可用選項的概觀，並介紹建立執行個體維護政策時需要考量的事項。

目錄

- [概要](#)
- [核心概念](#)
- [執行個體暖機期](#)
- [運作狀態檢查寬限期](#)
- [擴展 Auto Scaling 群組](#)
- [範例方案](#)

概要

當您為 Auto Scaling 群組建立執行個體維護政策時，該政策會影響導致執行個體被取代的 Amazon EC2 Auto Scaling 事件。這會導致在相同 Auto Scaling 群組中產生更一致的取代行為。此外，這也讓您可根據需求使群組的可用性或成本最佳化。

在主控台中可以使用下列組態選項：

- 終止前啟動 – 必須先佈建新執行個體，才能終止現有執行個體。對於重視可用性多於節省成本的應用程式來說，此做法是理想的選擇。
- 終止並啟動 – 在終止現有執行個體的同時佈建新執行個體。對於重視節省成本多於可用性的應用程式來說，此做法是理想的選擇。對於即使在取代執行個體時，也不應啟動超過目前可用容量的應用程式而言，這也是一個不錯的選擇。
- 自訂政策 – 此選項可讓您在取代執行個體時，根據想要的可用容量的自訂上下限範圍來設定政策。這種做法可協助您在成本和可用性之間取得適當的平衡。

Auto Scaling 群組預設為沒有執行個體維護政策，因此會以預設行為來回應執行個體維護事件。預設行為如下表所述。

執行個體維護事件預設行為

事件	Description	預設行為
運作狀態檢查失敗	在執行個體的運作狀態檢查失敗時自動發生。Amazon EC2	終止並啟動。

事件	Description	預設行為
	Auto Scaling 會取代運作狀態檢查失敗的執行個體。若要了解運作狀態檢查失敗的原因，請參閱 Auto Scaling 群組中的執行個體運作狀態檢查 。	
執行個體重新整理	在啟動執行個體重新整理時發生。視您的組態而定，執行個體重新整理可以一次取代一個執行個體、一次取代多個執行個體，或一次取代所有執行個體。如需詳細資訊，請參閱 使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體 。	終止並啟動。
執行個體生命週期上限	當執行個體達到您為 Auto Scaling 群組指定的執行個體生命週期上限時，則會自動發生。Amazon EC2 Auto Scaling 會取代達到執行個體生命週期上限的執行個體。如需詳細資訊，請參閱 根據執行個體生命週期上限來取代 Auto Scaling 執行個體 。	終止並啟動。

事件	Description	預設行為
重新平衡	如有導致群組不平衡的基礎變更，則會自動發生。Amazon EC2 Auto Scaling 會在下列情況下重新平衡群組： • 先前容量不足的可用區域復原，或者您從群組新增或移除可用區域。發生這種情況時，Auto Scaling 群組會嘗試在可用區域之間平衡自身。如需詳細資訊，請參閱重新平衡活動。 • 您在 Auto Scaling 群組上啟用容量重新平衡，它會在現有 Spot 執行個體因 Spot 執行個體的可用性變更而中斷之前，嘗試啟動新的 Spot 執行個體。如需詳細資訊，請參閱Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體。 • 您更新 Auto Scaling 群組，它會逐漸取代執行個體，以符合您在更新混合執行個體政策時選擇的新購買選項。如需詳細資訊，請參閱更新 Auto Scaling 群組。	終止前啟動。 Amazon EC2 Auto Scaling 可能會超出群組的大小限制，超出量最高可達其容量上限的 10%。但是，如果您使用的是容量重新平衡，則最多只能超出這些限制所需容量的 10%。

在下列情況下，Amazon EC2 Auto Scaling 將繼續預設為終止並啟動。因此，當其中一種情況發生時，群組的容量可能會小於執行個體維護政策的閾值下限。

- 例如執行個體因人為行動而意外終止時。Amazon EC2 Auto Scaling 立即取代不再執行的執行個體。如需詳細資訊，請參閱[Amazon EC2 運作狀態檢查](#)。

- 當 Amazon EC2 在 Amazon EC2 Auto Scaling 可以啟動替代執行個體之前，將執行個體做為排程事件的一部分而重新啟動、停止或淘汰。如需這些事件的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[執行個體的排程事件](#)。
- 當 Amazon EC2 Spot Service 啟動 Spot 執行個體中斷，之後 Spot 執行個體被強制終止時。

使用 Spot 執行個體時，如果您在 Auto Scaling 群組上啟用了容量重新平衡，則該執行個體可能已經有一個待執行個體，而該執行個體來自我們在起始 Spot 中斷之前啟動的另一個 Spot 集區。如需有關容量重新平衡運作的詳細資訊，請參閱 [Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體](#)。

但是，由於 Spot 執行個體無法保證會維持可用狀態，並且可能在兩分鐘的 Spot 執行個體中斷通知後終止，因此，如果執行個體在新執行個體啟動之前中斷，則可能會超過執行個體維護政策的閾值下限。

核心概念

在開始使用之前，請熟悉以下核心概念和術語：

所需容量

所需容量是建立 Auto Scaling 群組時的 Auto Scaling 群組容量。這也是群組未連接任何擴展條件時嘗試維持的容量。

執行個體維護政策

執行個體維護政策控制執行個體是否在執行個體維護事件終止現有執行個體之前先佈建一個執行個體。它也會決定 Auto Scaling 群組在同時取代多個執行個體時，可能會低於和超出所需容量的程度。

狀態良好最高百分比

狀態良好最高百分比是 Auto Scaling 群組在取代執行個體時可增加的所需容量百分比。它代表可以正常運作或待定的群組的百分比上限，以支援您的工作負載。在主控台中，您可以在使用終止前啟動選項或自訂政策選項時設定狀態良好最高百分比。有效值為 100-200%。

狀態良好最低百分比

狀態良好最低百分比是維持服務狀態、運作狀態良好，以及在取代執行個體時可用來支援工作負載的所需容量百分比。執行個體在成功完成其第一個運作狀態檢查並經過指定的暖機時間後，即視為運作狀態良好且可供使用。在主控台中，您可以在使用終止並啟動選項或自訂政策選項時設定狀態良好最低百分比。有效值為 0-100%。

Note

若要更快取代執行個體，您可以指定較低的狀態良好最低百分比。不過，如果執行中的執行個體運作狀態不夠良好，可能會降低可用性。我們建議您選取合理值，以在要取代多個執行個體的情況下維持可用性。

執行個體暖機期

如果執行個體在進入 InService 狀態後需要時間進行初始化，請為 Auto Scaling 群組啟用預設執行個體暖機期。使用預設執行個體暖機期，您便可以防止執行個體在準備就緒之前計入正常運作狀態百分比下限。這可確保 Amazon EC2 Auto Scaling 在終止現有執行個體之前，考慮到需要多長時間才能有足夠容量來支援工作負載。

另一項優點是，啟用預設的執行個體暖機期時，您可以改善用於動態擴展的 Amazon CloudWatch 指標。如果您的 Auto Scaling 群組有任何擴展政策，當群組橫向擴展時，會使用相同的預設暖機期，防止執行個體在完成初始化之前計入 CloudWatch 指標。

如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

運作狀態檢查寬限期

Amazon EC2 Auto Scaling 會根據您的 Auto Scaling 群組使用的運作狀態檢查狀態，判斷執行個體是否運作狀態良好。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

為確保這些運作狀態檢查能夠盡快啟動，請勿將群組的運作狀態檢查寬限期設定得太高，但要足以讓您的 Elastic Load Balancing 運作狀態檢查來判斷目標是否可用來處理要求。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。

擴展 Auto Scaling 群組

執行個體維護政策僅適用於執行個體維護事件，且不會防止群組手動或自動擴展。

如果 Auto Scaling 群組連接了擴展政策或排程動作，則可在執行個體維護事件發生時並行執行。在這種情況下，它們可以增減群組的所需容量，但只能在您定義的擴展限制內。如需這些限制的詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。

範例方案

在一般情況下，您的執行個體維護政策和所需容量可能如下所示：

- 狀態良好最低百分比 = 90%
- 狀態良好最高百分比 = 120%
- 所需容量 = 100

在任何執行個體維護事件期間，Auto Scaling 群組可能最少只有 90 個執行個體，最多只有 120 個執行個體。事件結束後，群組擁有的執行個體數量會恢復為 100 個。

當您將執行個體維護政策與具有暖集區的 Auto Scaling 群組搭配使用時，狀態良好最低和最高百分比會分別套用至 Auto Scaling 群組和暖集區。

例如，假設這是您的設定：

- 狀態良好最低百分比 = 90%
- 狀態良好最高百分比 = 120%
- 所需容量 = 100
- 暖集區大小 = 10

如果您啟動執行個體重新整理以回收群組的執行個體，Amazon EC2 Auto Scaling 會先取代 Auto Scaling 群組中的執行個體，然後取代暖集區中的執行個體。雖然 Amazon EC2 Auto Scaling 仍在取代 Auto Scaling 群組中的執行個體，但該群組可能最少只有 90 個執行個體，最多只有 120 個執行個體。完成群組取代程序後，Amazon EC2 Auto Scaling 便可以取代暖集區中的執行個體。發生這種情況時，暖集區可能最少只有 9 個執行個體，最多只有 12 個執行個體。

在 Auto Scaling 群組上設定執行個體維護政策

您可以在建立 Auto Scaling 群組時建立執行個體維護政策。您也可以為現有的群組建立此政策。

在 Auto Scaling 群組設定執行個體維護政策後，除非您想要覆寫執行個體維護，否則您不必再為執行個體重新整理功能指定狀態良好最低和最高百分比的參數。

在主控台中，Amazon EC2 Auto Scaling 提供可協助您快速入門的選項。

目錄

- [設定執行個體維護政策](#)
- [移除執行個體維護政策](#)

設定執行個體維護政策

若要在 Auto Scaling 群組上設定執行個體維護政策，請使用下列其中一種方法：

Console

在新群組設定執行個體維護政策 (主控台)

1. 依照 [使用啟動範本建立 Auto Scaling 群組](#) 的說明完成程序中的每個步驟，直到步驟 11。
2. 在設定群組大小和擴展政策上，為所需容量輸入要啟動之執行個體的初始數量。
3. 在擴展區段的擴展限制下，如果所需容量的新值大於所需容量下限和所需容量上限，則所需容量上限會自動增加到新的所需容量值。您可視需要變更這些限制。
4. 針對 Automatic scaling，選擇是否要建立目標追蹤擴展政策。您也可以在建 Auto Scaling 群組後建立此政策。

如果您選擇目標追蹤擴展政策，請依照 [建立目標追蹤擴展政策](#) 中的指示建立政策。

5. 在執行個體維護政策區段中，選擇其中一個可用選項：
 - 終止前啟動：必須先佈建新執行個體，才能終止現有執行個體。對於重視可用性多於節省成本的應用程式來說，這是理想的選擇。
 - 終止並啟動：在終止現有執行個體的同時佈建新執行個體。對於重視節省成本多於可用性的應用程式來說，這是理想的選擇。對於不應啟動超過目前可用容量的應用程式而言，這也是一個不錯的選擇。
 - 自訂政策：此選項可讓您根據想要在取代執行個體時可用容量的自訂上下限範圍來設定政策。這可協助您在成本和可用性之間取得適當的平衡。
6. 針對設定正常運作狀態百分比，請輸入下列一個或兩個欄位的值。啟用的欄位會根據您在上一個步驟所選選項而有所不同。
 - 下限：設定繼續取代執行個體所需的狀態良好最低百分比。
 - 上限：設定取代執行個體時可能的狀態良好最高百分比。
7. 展開根據所需容量檢視取代期間的容量區段，以確認如何在群組中套用下限和上限。使用的實際值取決於所需的容量值，此值會隨群組擴展而有所改變。
8. 繼續[使用啟動範本建立 Auto Scaling 群組](#)中的步驟。

AWS CLI

在新群組設定執行個體維護政策 (AWS CLI)

將 `--instance-maintenance-policy` 選項新增至 `create-auto-scaling-group` 命令。下列範例會針對名為 `my-asg` 的新 Auto Scaling 群組設定執行個體維護政策。

```
aws autoscaling create-auto-scaling-group \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --auto-scaling-group-name my-asg \
  --min-size 1 \
  --max-size 10 \
  --desired-capacity 5 \
  --default-instance-warmup 20 \
  --instance-maintenance-policy '{
    "MinHealthyPercentage": 90,
    "MaxHealthyPercentage": 120
  }' \
  --vpc-zone-identifier "subnet-5e6example,subnet-613example,subnet-c93example"
```

Console

在現有群組設定執行個體維護政策 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所在的 AWS 區域。
3. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在詳細資訊索引標籤上，選擇執行個體維護政策、編輯。
5. 若要在群組設定執行個體維護政策，選擇其中一個可用選項：
 - 終止前啟動：必須先佈建新執行個體，才能終止現有執行個體。對於重視可用性多於節省成本的應用程式來說，這是理想的選擇。
 - 終止並啟動：在終止現有執行個體的同時佈建新執行個體。對於重視節省成本多於可用性的應用程式來說，這是理想的選擇。對於不應啟動超過目前可用容量的應用程式而言，這也是一個不錯的選擇。
 - 自訂政策：此選項可讓您根據想要在取代執行個體時可用容量的自訂上下限範圍來設定政策。這可協助您在成本和可用性之間取得適當的平衡。
6. 針對設定正常運作狀態百分比，請輸入下列一個或兩個欄位的值。啟用的欄位會根據您在上一個步驟所選選項而有所不同。

- 下限：設定繼續取代執行個體所需的狀態良好最低百分比。
 - 上限：設定取代執行個體時可能的狀態良好最高百分比。
7. 展開根據所需容量檢視取代期間的容量區段，以確認如何在群組中套用下限和上限。使用的實際值取決於所需的容量值，此值會隨群組擴展而有所改變。
 8. 選擇更新。

AWS CLI

在現有群組上設定執行個體維護政策 (AWS CLI)

將 `--instance-maintenance-policy` 選項新增至 [update-auto-scaling-group](#) 命令。下列範例會針對指定 Auto Scaling 群組設定執行個體維護政策。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--instance-maintenance-policy '{  
    "MinHealthyPercentage": 90,  
    "MaxHealthyPercentage": 120  
}'
```

移除執行個體維護政策

如果您想要停止使用 Auto Scaling 群組的執行個體維護政策，您可以將其移除。

Console

移除執行個體維護政策 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所在的 AWS 區域。
3. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在詳細資訊索引標籤上，選擇執行個體維護政策、編輯。
5. 選擇無執行個體維護政策。
6. 選擇更新。

AWS CLI

移除執行個體維護政策 (AWS CLI)

將 `--instance-maintenance-policy` 選項新增至 [update-auto-scaling-group](#) 命令。下列範例為從指定 Auto Scaling 群組移除執行個體維護政策。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --instance-maintenance-policy '{
    "MinHealthyPercentage": -1,
    "MaxHealthyPercentage": -1
  }'
```

Amazon EC2 Auto Scaling lifecycle hook

Amazon EC2 Auto Scaling 提供將 lifecycle hook 新增到 Auto Scaling 群組的能力。這些掛鉤可讓您建立了解 Auto Scaling 執行個體生命週期中之事件的解決方案，然後在對應的生命週期事件發生時對執行個體執行自訂動作。lifecycle hook 提供指定的時間 (預設為一小時)，以便在執行個體轉換到下一個狀態之前等待動作完成。

作為結合使用 lifecycle hook 與 Auto Scaling 執行個體的範例：

- 水平擴展事件發生時，新啟動的執行個體會完成啟動序列，並轉換到等待狀態。當執行個體處於等待狀態時，其會執行指令碼以下載和安裝應用程式所需的軟體套件，確保執行個體在開始接收流量之前已經完全準備妥當。當指令碼完成軟體的安裝後，其會傳送 `complete-lifecycle-action` 命令以繼續。
- 發生縮減事件時，lifecycle hook 會在執行個體終止前將其暫停，並使用 Amazon EventBridge 傳送通知給您。當執行個體處於等待狀態時，您可以在執行個體完全終止之前叫用 AWS Lambda 函數或連線到執行個體以下載日誌或其他資料。

lifecycle hook 的常見用途是控制何時使用 Elastic Load Balancing 註冊執行個體。透過將啟動 lifecycle hook 新增至 Auto Scaling 群組，您可以確保引導指令碼已成功完成，且執行個體上的應用程式已準備好接受流量，然後在 lifecycle hook 結束時向負載平衡器註冊執行個體。

目錄

- [lifecycle hook 可用性](#)
- [lifecycle hook 的考量與限制](#)
- [相關資源](#)

- [Auto Scaling 群組中的生命週期關聯運作方式](#)
- [準備將 lifecycle hook 新增至您的 Auto Scaling 群組](#)
- [使用執行個體生命週期政策控制執行個體保留](#)
- [透過執行個體中繼資料擷取目標生命週期狀態](#)
- [將生命週期掛鉤新增至 Auto Scaling 群組](#)
- [在 Auto Scaling 群組中完成生命週期動作](#)
- [教學課程：使用資料指令碼和執行個體中繼資料來擷取生命週期狀態](#)
- [教學課程：設定叫用 Lambda 函數的 lifecycle hook](#)

lifecycle hook 可用性

下表列出可用於各種案例的 lifecycle hook。

事件	執行個體啟動 或終止 ¹	執行個體生命週期上限 ：取代執行個體	執行個體重新整理 ：取代執行個體	容量重新平衡 ：取代執行個體	暖集區 ：進入和離開暖集區的執行個體
執行個體啟動	✓	✓	✓	✓	✓
執行個體終止	✓	✓	✓	✓	✓

¹ 適用於所有啟動和終止，無論是自動還是手動啟動，例如當您呼叫 `SetDesiredCapacity` 或 `TerminateInstanceInAutoScalingGroup` 操作時。在您連接或分開執行個體、將執行個體移入和移出待命模式或使用強制刪除選項刪除群組時不適用。

lifecycle hook 的考量與限制

配合使用生命週期關聯時，請記住下列考量事項和限制：

- Amazon EC2 Auto Scaling 提供自己的生命週期，協助管理 Auto Scaling 群組。此生命週期與其他 EC2 執行個體的生命週期不同。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 執行個體生命週期](#)。暖集區中的執行個體也有自己的生命週期，如 [暖集區中的執行個體生命週期狀態轉換](#) 中所述。
- 根據預設，終止生命週期掛鉤會盡力運作。如果終止生命週期關聯逾時或遭到捨棄，Amazon EC2 Auto Scaling 會繼續立即終止執行個體。您可以結合終止生命週期關聯與執行個體保留的執行個體生命週期政策。如需詳細資訊，請參閱[使用執行個體生命週期政策控制執行個體保留](#)。

- 您可以使用含有 Spot 執行個體的 lifecycle hook，不過，如果執行個體的容量已無法再使用，lifecycle hook 無法防止執行個體終止，這種情況隨時都可能發生，並會發出兩分鐘的中斷通知。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [Spot 執行個體中斷](#)。但是，您可以啟用容量重新平衡功能，藉此主動取代從 Amazon EC2 Spot 服務收到重新平衡建議的 Spot 執行個體；這是在 Spot 執行個體處於較高中斷風險時傳送的訊號。如需詳細資訊，請參閱[Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體](#)。
- 執行個體可以在有限的期間內維持於等待狀態。lifecycle hook 的預設逾時為 1 小時 (活動訊號逾時)。此外，還有一個全域逾時，指定您可將執行個體保留在等待狀態的最長時間。全域逾時為 48 小時或活動訊號逾時的 100 倍，以較小值為準。
- 系統可能捨棄或繼續 lifecycle hook 的結果。如果某個執行個體正在啟動，繼續則表示您的動作已成功，而且 Amazon EC2 Auto Scaling 可將執行個體投入服務。否則，捨棄則表示您的自訂動作沒有成功執行，且我們可能會終止並取代該執行個體。如果某個執行個體正在終止，捨棄和繼續都會允許該執行個體終止。不過，放棄會停止任何剩餘的動作，如其他 lifecycle hook，而繼續允許讓任何其他 lifecycle hook 來完成。
- Amazon EC2 Auto Scaling 會限制 lifecycle hook 持續失敗時允許執行個體啟動的速率，因此請務必測試並修復生命週期動作中的任何永久性錯誤。
- 使用 AWS CLI CloudFormation 或 SDK 建立和更新生命週期掛鉤，提供從 建立生命週期掛鉤時無法使用的選項 AWS 管理主控台。例如，用於指定 SNS 主題或 SQS 佇列的 ARN 欄位不會顯示在主控台中，因為 Amazon EC2 Auto Scaling 已向 Amazon EventBridge 傳送事件。您可以視需要篩選這些事件並重新導向至 Lambda、Amazon SNS 和 Amazon SQS 等 AWS 服務。
- 您可以在建立 Auto Scaling 群組時，使用 呼叫 [CreateAutoScalingGroup](#) API 或 SDK AWS CLI CloudFormation，將多個生命週期掛鉤新增至該群組。但是，如果指定，每個掛鉤必須具有相同的通知目標和 IAM 角色。若要建立具有不同通知目標和不同角色的 lifecycle hook，請在對 [PutLifecycleHook](#) API 的單獨叫用中一次建立一個 lifecycle hook。
- 如果您為執行個體啟動新增 lifecycle hook，運作狀態檢查寬限期會在執行個體到達 InService 狀態時開始。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。

擴展考量

- 動態擴展政策可根據跨多個執行個體彙總的 CloudWatch 指標資料 (例如 CPU 和網路 I/O) 縮減和橫向擴展。橫向擴展時，Amazon EC2 Auto Scaling 不會立即將新執行個體計入 Auto Scaling 群組的彙總執行個體指標。它會等待執行個體到達 InService 狀態且執行個體暖機完成為止。如需詳細資訊，請參閱預設執行個體暖機主題中的 [擴展效能考量](#)。
- 在縮減時，彙總的執行個體指標可能不會立即反映終止執行個體的移除情況。Amazon EC2 Auto Scaling 終止工作流程開始後不久，正在終止的執行個體會停止計入群組的彙總執行個體指標。

- 在大多數情況下，當調用生命週期關聯時，簡易擴展政策引起的擴展活動會暫停，直至生命週期動作完成且冷卻時間已過。設定較長的冷卻時間間隔代表需要更多的時間才能恢復擴展。如需詳細資訊，請參閱冷卻時間主題中的 [生命週期關聯可能會導致額外延遲](#)。一般而言，如果您可以改用步進擴展或目標追蹤擴展政策，建議您不要使用簡易擴展政策。

相關資源

如需簡介影片，請參閱 YouTube 上的 [AWS re:Invent 2018：透過 Amazon EC2 Auto Scaling 輕鬆管理容量](#)。

我們提供幾個 JSON 和 YAML 範本程式碼片段，您可以用來了解如何在 CloudFormation 堆疊範本中宣告生命週期關聯。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-autoscaling-lifecyclehook.html> 中的 `AWS::AutoScaling::LifecycleHook` 參考資料。

您也可以造訪我們的 [GitHub 儲存庫](#)，下載 lifecycle hook 的範例範本和使用者資料指令碼。

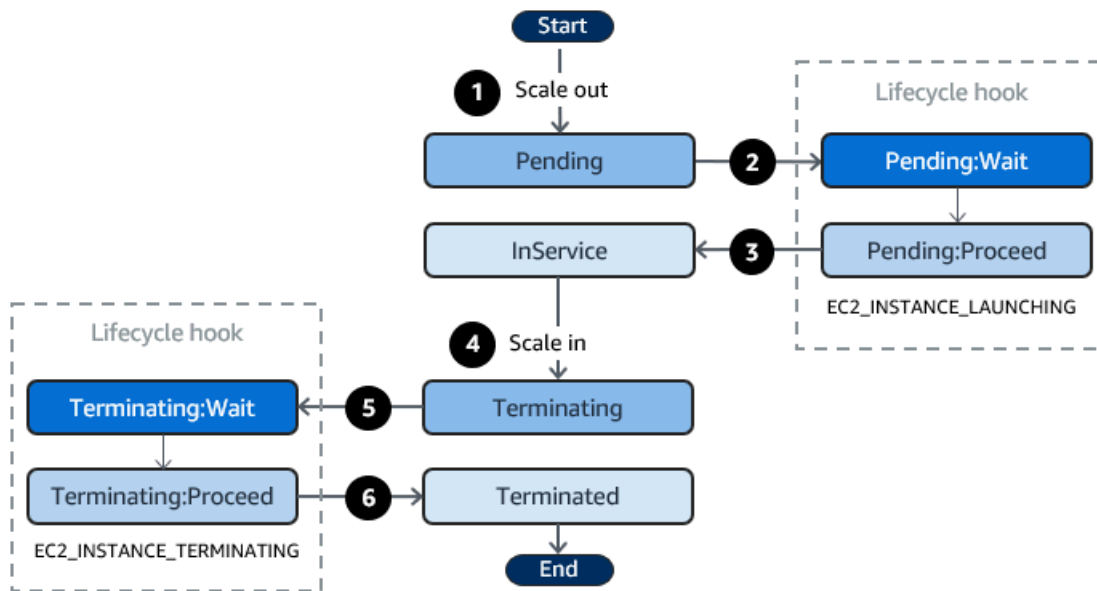
如需生命週期關聯的使用範例，請參閱下列部落格文章。

- 《[使用 Lambda 和 Amazon EC2 執行命令為擴展的執行個體建立備份系統](#)》
- 《[在終止 EC2 Auto Scaling 執行個體之前執行程式碼](#)》。

Auto Scaling 群組中的生命週期關聯運作方式

從啟動到終止，Amazon EC2 執行個體會轉換到不同狀態。您可以為 Auto Scaling 群組建立自訂動作，以便在執行個體因 lifecycle hook 而轉換到等待狀態時執行該動作。

下圖顯示當您使用生命週期掛鉤向外擴展和向內擴展時 Auto Scaling 執行個體狀態之間的轉換。



如前面的圖表所示：

1. Auto Scaling 群組會回應水平擴展事件，並開始啟動執行個體。
2. lifecycle hook 使執行個體進入等待狀態 (Pending:Wait)，接著會執行自訂動作。

在您完成生命週期動作前，或逾時期限結束前，執行個體都保持在等待狀態。根據預設，執行個體會維持等待狀態一小時的時間，然後 Auto Scaling 群組就會繼續啟動程序 (Pending:Proceed)。如果您需要更多的時間，您可以利用活動訊號記錄重新啟動逾時期限。如果在自訂動作完成時且逾時期限尚未到期之前完成生命週期動作，則該期限會結束，Auto Scaling 群組會繼續啟動程序。

3. 執行個體會進入 InService 狀態，運作狀態檢查寬限期就會開始。但是，在執行個體達到 InService 狀態之前，如果 Auto Scaling 群組與 Elastic Load Balancing 負載平衡器建立關聯，則會向負載平衡器註冊執行個體，而負載平衡器會開始檢查其運作狀態。運作狀態檢查寬限期結束後，Amazon EC2 Auto Scaling 會開始檢查執行個體的運作狀態。
4. Auto Scaling 群組會回應縮減事件，並開始終止執行個體。如果 Auto Scaling 群組與 Elastic Load Balancing 搭配使用，則會先將終止的執行個體從負載平衡器中取消註冊。如果已啟用負載平衡器的連接耗盡，執行個體會停止接受新的連接，並等待現有連接耗盡，再完成取消註冊程序。
5. lifecycle hook 使執行個體進入等待狀態 (Terminating:Wait)，接著會執行自訂動作。

在您完成生命週期動作前，或逾時期限結束 (預設為一小時) 前，執行個體都保持在等待狀態。完成 lifecycle hook 或逾時期限到期後，執行個體會轉換到下一個狀態 (Terminating:Proceed)。

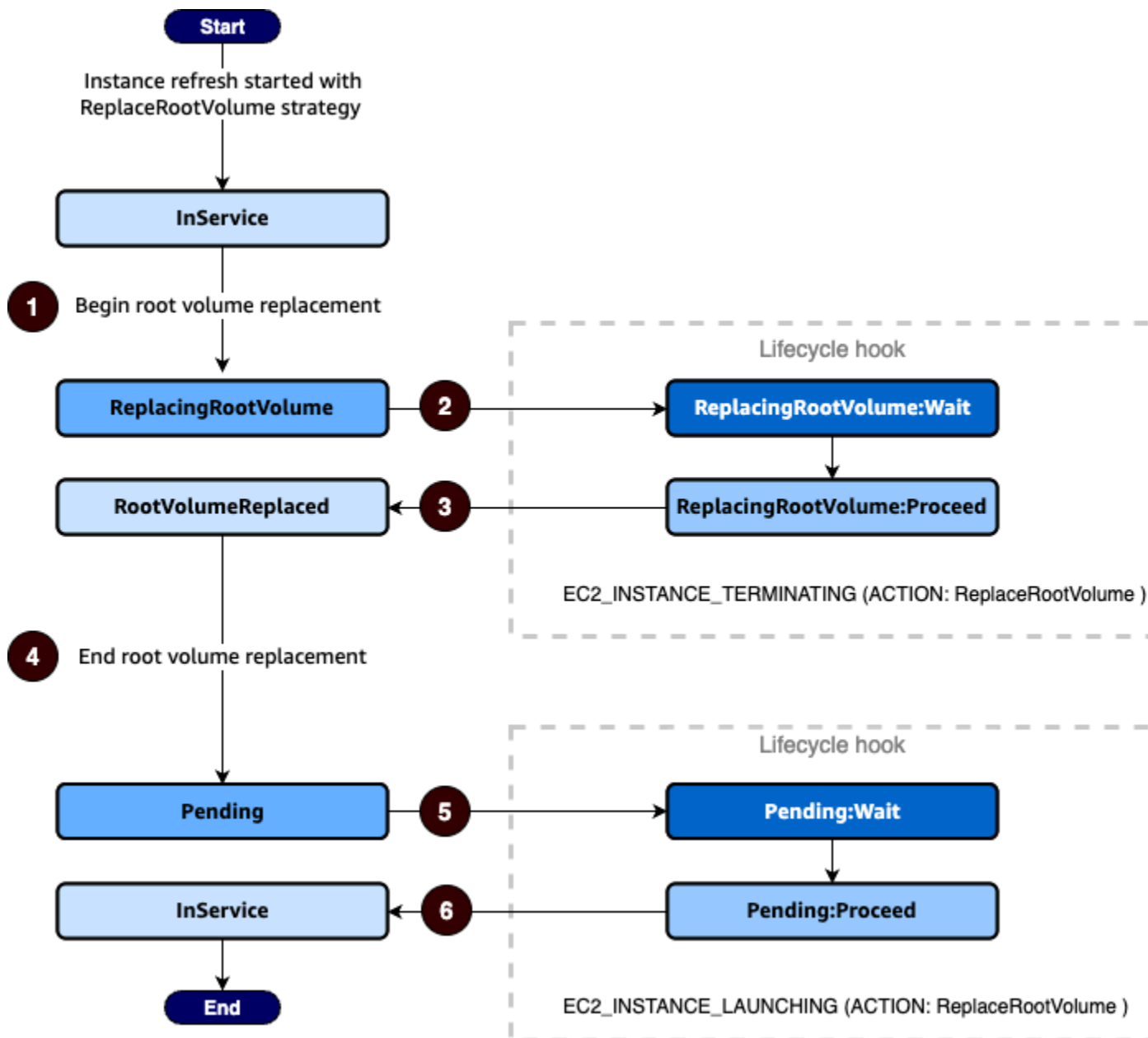
6. 執行個體已終止。

⚠ Important

暖集區中的執行個體也有自己的生命週期及對應的等待狀態，如 [暖集區中的執行個體生命週期狀態轉換](#) 中所述。

正在進行根磁碟區取代之執行個體的生命週期狀態轉換

下圖顯示當您使用生命週期掛鉤取代根磁碟區時Auto Scaling 執行個體狀態之間的轉換：



如前面的圖表所示：

1. Auto Scaling 群組會回應執行個體重新整理，並選取執行個體以取代根磁碟區。執行個體進入 `ReplacingRootVolume` 狀態。如果執行個體已向負載平衡器註冊，則會從負載平衡器取消註冊。
2. lifecycle hook 使執行個體進入等待狀態 (`ReplacingRootVolume:Wait`)，接著會執行自訂動作。在您完成生命週期動作前，或逾時期限結束前，執行個體都保持在等待狀態。如果您在自訂動作完成且逾時期間尚未過期時完成生命週期動作，則期間會結束，Auto Scaling 群組會繼續根磁碟區替換程序。
3. 執行個體完成其根磁碟區取代並進入 `RootVolumeReplaced` 狀態。
4. 執行個體進入 `Pending` 狀態。
5. lifecycle hook 使執行個體進入等待狀態 (`Pending:Wait`)，接著會執行自訂動作。執行個體會保持等待狀態，直到您完成生命週期動作，或直到逾時期間結束為止。完成 lifecycle hook 或逾時期限到期後，執行個體會轉換到下一個狀態 (`Pending:Proceed`)。
6. 執行個體進入 `InService` 狀態。不過，在執行個體達到 `InService` 狀態之前，如果 Auto Scaling 群組與 Elastic Load Balancing 負載平衡器相關聯，則會向負載平衡器註冊執行個體。

準備將 lifecycle hook 新增至您的 Auto Scaling 群組

將 lifecycle hook 新增至 Auto Scaling 群組之前，請確保正確設定使用者資料指令碼或通知目標。

- 若要在執行個體啟動時使用使用者資料指令碼以在執行個體上執行自訂動作，您不需要設定通知目標。但是，您必須建立好指定使用者資料指令碼的啟動範本或啟動組態，並將其與 Auto Scaling 群組相互關聯。如需使用者資料指令碼的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[在啟動時在 Linux 執行個體上執行命令](#)。
- 若要在生命週期動作完成時向 Amazon EC2 Auto Scaling 發出訊號，則必須在指令碼中加入 [CompleteLifecycleAction](#) API 呼叫，且您必須手動建立 IAM 角色，而該角色的政策允許 Auto Scaling 執行個體呼叫此 API。您的啟動範本或啟動組態必須使用啟動時即連接至 Amazon EC2 執行個體的 IAM 執行個體設定檔來指定此角色。如需詳細資訊，請參閱[在 Auto Scaling 群組中完成生命週期動作](#)及[在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)。
- 若要允許 Lambda 在生命週期動作完成時向 Amazon EC2 Auto Scaling 發出訊號，您必須在函數程式碼中加入 [CompleteLifecycleAction](#) API 呼叫。您還必須將 IAM 政策連接至函數的執行角色，該執行角色能授予 Lambda 完成生命週期動作的許可。如需詳細資訊，請參閱[教學課程：設定叫用 Lambda 函數的 lifecycle hook](#)。
- 若要使用 Amazon SNS 或 Amazon SQS 等服務執行自訂動作，您必須已建立 SNS 主題或 SQS 佇列，並準備好其 Amazon Resource Name (ARN)。您還必須建立好 IAM 角色，允許 Amazon EC2 Auto Scaling 存取您的 SNS 主題或 SQS 目標，並準備好其 ARN。如需詳細資訊，請參閱[設定生命週期通知的通知目標](#)。

 Note

預設情況下，您在主控台中新增 lifecycle hook 時，Amazon EC2 Auto Scaling 便會向 Amazon EventBridge 傳送生命週期事件通知。使用 EventBridge 或使用者資料指令碼是建議的最佳實務。若要建立直接將通知傳送到 Amazon SNS、Amazon SQS 或的生命週期關聯 AWS Lambda，請使用 AWS CLI、AWS CloudFormation 或 SDK 來新增生命週期關聯。

設定生命週期通知的通知目標

您可以將 lifecycle hook 新增至 Auto Scaling 群組，以便在執行個體進入等待狀態時執行自訂動作。您可以根據偏好的開發方法，選擇目標服務來執行這些動作。

實作生命週期關聯通知目標的方法有四種：

- Amazon EventBridge — 接收通知並執行您想要的動作。
- Amazon Simple Notification Service (Amazon SNS) — 建立發佈通知的主題。用戶端可以訂閱 SNS 主題，並使用支援的通訊協定接收發佈的訊息。
- Amazon Simple Queue Service (Amazon SQS) — 透過輪詢模型交換訊息。
- AWS Lambda — 叫用執行所需動作的 Lambda 函數。

最佳實務建議您使用 EventBridge。傳送至 Amazon SNS 和 Amazon SQS 的通知包含的資訊與 Amazon EC2 Auto Scaling 傳送至 EventBridge 的通知相同。在 EventBridge 之前，標準實務是將通知傳送至 SNS 或 SQS，並將其他服務與 SNS 或 SQS 整合，以執行程式設計動作。如今，EventBridge 為您提供更多選項，可讓您選擇以哪些服務為目標，並能更輕鬆地透過無伺服器架構處理事件。

請記住，如果啟動範本或啟動組態中的使用者資料指令碼會在執行個體啟動時對其進行設定，則您無需接收通知，即可在執行個體上執行自訂動作。

下列步驟介紹如何設定通知目標。

目錄

- [使用 EventBridge 將通知路由至 Lambda](#)
- [使用 Amazon SNS 接收通知](#)
- [使用 Amazon SQS 接收通知](#)

- [將通知 AWS Lambda 直接路由到](#)
- [通知訊息範例](#)

Important

搭配 lifecycle hook 使用的 EventBridge 規則、Lambda 函數、Amazon SNS 主題和 Amazon SQS 佇列的所在區域必須永遠與 Auto Scaling 群組建立所在的區域相同。

使用 EventBridge 將通知路由至 Lambda

您可以將 EventBridge 規則設定為在執行個體進入等待狀態時叫用 Lambda 函數。Amazon EC2 Auto Scaling 會向 EventBridge 傳送關於正在啟動或終止的執行個體的生命週期事件通知，以及可用於控制生命週期動作的權杖。如需這些事件的範例，請參閱 [Amazon EC2 Auto Scaling 事件參考](#)。

Note

當您使用 AWS 管理主控台 建立事件規則時，主控台會自動新增必要的 IAM 許可，以授予 EventBridge 呼叫 Lambda 函數的許可。如果您使用 AWS CLI 建立事件規則，則需要明確地授予此許可。

如需有關如何在 EventBridge 主控台中建立事件規則的資訊，請參閱《Amazon EventBridge 使用者指南》中的「[建立對事件做出反應的 Amazon EventBridge 規則](#)」。

– 或 –

如需關於主控台使用者的簡介教學課程，請參閱 [教學課程：設定叫用 Lambda 函數的 lifecycle hook](#)。此教學課程能讓您了解如何建立簡單的 Lambda 函數，以接聽啟動事件並將其寫入 CloudWatch Logs 日誌。

建立叫用 Lambda 函數的 EventBridge 規則

1. 使用 [Lambda 主控台](#) 建立 Lambda 函數，及記下 Amazon Resource Name (ARN)。例如 `arn:aws:lambda:region:123456789012:function:my-function`。您需要 ARN 來建立 EventBridge 目標。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 入門](#)。
2. 若要建立符合執行個體啟動之事件的規則，請使用下列 [put-rule](#) 命令。

```
aws events put-rule --name my-rule --event-pattern file://pattern.json --state  
ENABLED
```

以下範例展示適用於執行個體啟動生命週期動作的 `pattern.json`。將 `##` 顯示文字取代為您的 Auto Scaling 群組名稱。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "AutoScalingGroupName": [ "my-asg" ]
  }
}
```

如果命令執行成功，EventBridge 會使用規則的 ARN 來回應。請記住 ARN。您需要在步驟 4 輸入此名稱。

若要建立符合其他事件的規則，請修改事件模式。如需詳細資訊，請參閱[使用 EventBridge 處理 Auto Scaling 事件](#)。

- 若要指定要用作規則目標的 Lambda 函數，請使用以下 `put-target` 命令。

```
aws events put-targets --rule my-rule --targets
  Id=1,Arn=arn:aws:lambda:region:123456789012:function:my-function
```

在上述命令中，`my-rule` 是您在步驟 2 中為規則指定的名稱，而 `Arn` 參數的值則是您在步驟 1 中建立的函數 ARN。

- 若要新增允許規則呼叫目標 Lambda 函數的許可，請使用以下 Lambda `add-permission` 命令。此命令會信任 EventBridge 服務委託人 (`events.amazonaws.com`)，並根據指定的規則來設定許可的範圍。

```
aws lambda add-permission --function-name my-function --statement-id my-unique-id \
  --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
  arn:aws:events:region:123456789012:rule/my-rule
```

在上述命令中：

- `my-function` 是您希望讓規則作為目標使用的 Lambda 函數名稱。
- `my-unique-id` 是您定義來描述 Lambda 函數政策陳述式的唯一識別符。
- `source-arn` 為 EventBridge 規則的 ARN。

如果此命令成功執行，您會收到類似如下的輸出。

```
{
  "Statement": "{\"Sid\":\"my-unique-id\",
    \"Effect\":\"Allow\",
    \"Principal\":{\"Service\":\"events.amazonaws.com\"},
    \"Action\":\"lambda:InvokeFunction\",
    \"Resource\":\"arn:aws:lambda:us-west-2:123456789012:function:my-function\",
    \"Condition\":
      {\"ArnLike\":
        {\"AWS:SourceArn\":
          \"arn:aws:events:us-west-2:123456789012:rule/my-rule\"}}}"
}
```

Statement 值是陳述式的 JSON 字串版本，且已新增至 Lambda 函數政策。

5. 當您遵照這些指示後，請繼續進行 [將生命週期掛鉤新增至 Auto Scaling 群組](#) 步驟。

使用 Amazon SNS 接收通知

您可以使用 Amazon SNS 設定通知目標 (SNS 主題)，在生命週期動作發生時接收通知。然後，Amazon SNS 會將通知傳送給訂閱的收件人。在確認訂閱前，都不會將發佈至主題的通知傳送給收件人。

使用 Amazon SNS 設定通知

1. 使用 [Amazon SNS 主控台](#) 或下列 `create-topic` 命令建立 Amazon SNS 主題。請確認此主題與您使用的 Auto Scaling 群組位於同一個區域。如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的 [Amazon SNS 入門](#)。

```
aws sns create-topic --name my-sns-topic
```

2. 請記下主題 Amazon Resource Name (ARN)，例如 `arn:aws:sns:region:123456789012:my-sns-topic`。您需要此資訊來建立 lifecycle hook。
3. 建立 IAM 服務角色，讓 Amazon EC2 Auto Scaling 能夠存取 Amazon SNS 通知目標。

讓 Amazon EC2 Auto Scaling 存取 SNS 主題。

- a. 前往網址 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
 - b. 在左側導覽窗格中，選擇 Roles (角色)。
 - c. 選擇建立角色。
 - d. 對於 Select trusted entity (選取信任的實體) 區段，選擇 AWS service (AWS 服務)。
 - e. 針對您的使用案例，在 Use cases for other AWS services (其他 服務的使用案例) 下，選擇 EC2 Auto Scaling，然後選擇 EC2 Auto Scaling Notification Access (EC2 Auto Scaling 通知存取)。
 - f. 選擇 Next (下一步) 兩次，前往 Name, review, and create (命名、檢閱和建立) 頁面。
 - g. 在 Role name (角色名稱) 欄位中，輸入角色名稱 (例如：**my-notification-role**)，然後選擇 Create role (建立角色)。
 - h. 在 Roles (角色) 頁面上，選擇您剛建立的角色，以開啟 Summary (摘要) 頁面。請記下角色 ARN。例如 `arn:aws:iam::123456789012:role/my-notification-role`。您需要此資訊來建立 lifecycle hook。
4. 當您遵照這些指示後，請繼續進行下一步：[新增 lifecycle hook \(AWS CLI\)](#)。

使用 Amazon SQS 接收通知

您可以在生命週期動作發生時，使用 Amazon SQS 設定通知目標來接收訊息。佇列取用者接著必須輪詢 SQS 佇列，才能對這些通知執行動作。

Important

FIFO 佇列與 lifecycle hook 不相容。

使用 Amazon SQS 設定通知

1. 使用 [Amazon SQS 主控台](#) 建立 Amazon SQS 佇列。確定佇列與您使用的 Auto Scaling 群組位於相同區域。如需詳細資訊，請參閱《Amazon Simple Queue Service 開發人員指南》中的 [Amazon SQS 入門](#)。
2. 記下佇列 ARN，例如 `arn:aws:sqs:us-west-2:123456789012:my-sqs-queue`。您需要此資訊來建立 lifecycle hook。
3. 建立 IAM 服務角色，讓 Amazon EC2 Auto Scaling 存取 Amazon SQS 通知目標。

讓 Amazon EC2 Auto Scaling 存取 SQS 佇列

- a. 前往網址 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
 - b. 在左側導覽窗格中，選擇 Roles (角色)。
 - c. 選擇建立角色。
 - d. 對於 Select trusted entity (選取信任的實體) 區段，選擇 AWS service (AWS 服務)。
 - e. 針對您的使用案例，在 Use cases for other AWS services (其他服務的使用案例) 下，選擇 EC2 Auto Scaling，然後選擇 EC2 Auto Scaling Notification Access (EC2 Auto Scaling 通知存取)。
 - f. 選擇 Next (下一步) 兩次，前往 Name, review, and create (命名、檢閱和建立) 頁面。
 - g. 在 Role name (角色名稱) 欄位中，輸入角色名稱 (例如：**my-notification-role**)，然後選擇 Create role (建立角色)。
 - h. 在 Roles (角色) 頁面上，選擇您剛建立的角色，以開啟 Summary (摘要) 頁面。請記下角色 ARN。例如 `arn:aws:iam::123456789012:role/my-notification-role`。您需要此資訊來建立 lifecycle hook。
4. 當您遵照這些指示後，請繼續進行下一步：[新增 lifecycle hook \(AWS CLI\)](#)。

將通知 AWS Lambda 直接路由到

生命週期動作發生時，您可以使用 Lambda 函數做為通知目標。

將通知 AWS Lambda 直接路由到

1. 開啟 Lambda 主控台中的 [Functions \(函數\) 頁面](#)。
2. 選擇您想要的 Lambda 函數。

如果您想要建立新的 Lambda 函數，請參閱 [建立 Lambda 函式](#)

3. 選擇組態索引標籤，然後選擇許可。
4. 向下捲動至 Resource-based policy (資源型政策)，然後選擇 Add permissions (新增許可)。資源型政策可用於向政策中所指定的委託人授予叫用函數的許可。在這種情況下，委託人是與 Auto Scaling 群組相關聯的 [Amazon EC2 Auto Scaling 服務連結角色](#)。
5. 在 Policy statement (政策陳述式) 區段中，設定您的許可：
 - a. 選擇 AWS 帳戶。
 - b. 在 Principal (委託人) 中，輸入呼叫服務連結角色的 ARN，例如 `arn:aws:iam::<aws-account-id>:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling`。

- c. 在 Action (動作) 中，選擇 `lambda:InvokeFunction`。
 - d. 在 Statement ID (陳述式 ID) 中，請輸入唯一的陳述式 ID，例如 **`AllowInvokeByAutoScaling`**。
 - e. 選擇儲存。
6. 當您遵照這些指示後，請繼續進行下一步：[新增 lifecycle hook \(AWS CLI\)](#)。

通知訊息範例

本節提供 Amazon SNS、Amazon SQS 和 的通知範例 AWS Lambda。

當執行個體處於等待狀態時，訊息會發佈至 Amazon SNS、Amazon SQS 和 AWS Lambda 通知目標。

訊息包括下列資訊：

- `Origin` — EC2 執行個體的來源位置。
- `Destination` — EC2 執行個體要前往的地方。
- `LifecycleActionToken`：生命週期動作字符。
- `AccountId` — AWS 帳戶 ID。
- `AutoScalingGroupName`：Auto Scaling 群組的名稱。
- `LifecycleHookName`：lifecycle hook 的名稱。
- `EC2InstanceId`：EC2 執行個體的 ID。
- `LifecycleTransition`：lifecycle hook 類型。
- `NotificationMetadata`：通知中繼資料。

以下是通知訊息範例。

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:36:26.533Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Origin: EC2
Destination: AutoScalingGroup
LifecycleActionToken: 71514b9d-6a40-4b26-8523-05e7ee35fa40
AccountId: 123456789012
AutoScalingGroupName: my-asg
LifecycleHookName: my-hook
```

```
EC2InstanceId: i-0598c7d356eba48d7
LifecycleTransition: autoscaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata: hook message metadata
```

測試通知訊息範例

首次新增 lifecycle hook 時，測試通知訊息會發佈到通知目標。以下是測試通知訊息範例。

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:35:52.359Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Event: autoscaling:TEST_NOTIFICATION
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg
```

Note

如需從 Amazon EC2 Auto Scaling 傳遞到 EventBridge 的事件範例，請參閱 [Amazon EC2 Auto Scaling 事件參考](#)。

使用執行個體生命週期政策控制執行個體保留

執行個體生命週期政策可在捨棄終止生命週期動作時提供 Amazon EC2 Auto Scaling 終止的保護。與單獨使用生命週期掛鉤不同，執行個體生命週期政策旨在確保執行個體在正常關機程序未成功完成時移至保留狀態。

何時使用執行個體生命週期政策

當應用程式的正常關閉不是選用的，但強制和失敗的關閉需要手動介入時，請使用執行個體生命週期政策。常用案例包括：

- 必須在終止前完成資料持久性的狀態應用程式。
- 需要延長耗盡期間的應用程式，可能超過生命週期關聯逾時上限 48 小時。
- 當清除失敗或不完整時，處理敏感資料的工作負載可能會導致資料遺失或損毀。
- 任務關鍵服務，其中突然關機會對可用性造成影響。

如需如何正常處理執行個體終止的詳細資訊，請參閱 [設計您的應用程式以正常處理執行個體終止](#)。

執行個體生命週期政策如何與終止生命週期關聯搭配使用

執行個體生命週期政策可與終止生命週期掛鉤搭配使用，而非取代。程序遵循數個階段：

1. 執行終止生命週期動作。當 Amazon EC2 Auto Scaling 選取要終止的執行個體時，系統會叫用您的終止生命週期關聯，且執行個體會進入 `Terminating:Wait` 狀態以開始執行終止生命週期動作。
2. 正常關機嘗試開始。您的應用程式，無論是在執行個體上執行或透過控制平面執行，都會收到終止生命週期動作通知，並開始正常的關閉程序，例如耗盡連線、完成進行中的工作或傳輸資料。
3. 終止生命週期動作完成。終止生命週期動作可以使用 `CONTINUE` 或 `ABANDON` 結果完成。
4. 執行個體生命週期政策會評估情況。如果未設定執行個體生命週期政策，即使終止生命週期動作已完成 `ABANDON`，執行個體仍會立即終止。在執行個體生命週期政策設定為在 上保留執行個體的情況下 `TerminateHookAbandon`，如果終止生命週期動作已完成 `ABANDON`，執行個體會移至保留狀態。
5. 保留的執行個體會等待手動動作。處於保留狀態的執行個體會持續產生標準 Amazon EC2 費用。這些執行個體不會計入 Auto Scaling 群組所需的容量，因此 Auto Scaling 會啟動替換執行個體以維持所需的大小。Auto Scaling 功能，例如執行個體重新整理和執行個體生命週期上限，也會忽略保留的執行個體。這可讓您在手動終止執行個體之前，手動完成清除程序、復原資料或調查自動關機失敗的原因。
6. 手動終止發生。在保留執行個體上完成必要的動作後，您需要呼叫 `TerminateInstanceInAutoScalingGroup` API 來終止執行個體。

設定執行個體保留

設定 Amazon EC2 Auto Scaling 群組，以在終止生命週期動作失敗時保留執行個體。

若要在 Auto Scaling 群組中使用執行個體生命週期政策，您還必須設定終止生命週期關聯。如果您設定執行個體生命週期政策，但沒有任何終止生命週期關聯，則政策沒有作用。只有在捨棄終止生命週期動作時，才會套用執行個體生命週期政策，而不是當它們成功完成 `CONTINUE` 結果時。

執行個體生命週期政策使用保留觸發來判斷何時保留執行個體。`TerminateHookAbandon` 觸發會在幾個案例中造成保留：

- 當您使用 `ABANDON` 結果明確呼叫 [CompleteLifecycleAction](#) API 時。
- 當具有預設結果的終止生命週期動作因為達到活動訊號逾時而不接收活動訊號而 `ABANDON` 逾時時。

- 當在具有預設結果的終止生命週期動作上達到全域逾時時ABANDON，即 48 小時或活動訊號逾時的 100 倍，以較小者為準

Console

設定執行個體保留

1. 開啟 Amazon EC2 Auto Scaling 主控台
2. 建立 Auto Scaling 群組（執行個體生命週期政策預設為終止）
3. 前往 Auto Scaling 群組詳細資訊頁面，然後選擇執行個體管理索引標籤
4. 在生命週期關聯的執行個體生命週期政策中，選擇保留
5. 使用 建立您的終止生命週期關聯：
 - 生命週期轉換設定為執行個體終止
 - 預設結果設定為放棄

AWS CLI

設定執行個體保留

使用 [create-auto-scaling-group](#) 命令搭配執行個體生命週期政策：

```
aws autoscaling create-auto-scaling-group \  
--auto-scaling-group-name my-asg \  
--launch-template LaunchTemplateName=my-template,Version='$Latest' \  
--min-size 1 \  
--max-size 3 \  
--desired-capacity 2 \  
--vpc-zone-identifier subnet-12345678 \  
--instance-lifecycle-policy file:///lifecycle-policy.json
```

lifecycle-policy.json 的內容：

```
{  
  "RetentionTriggers": {  
    "TerminateHookAbandon": "retain"  
  }  
}
```

新增終止 lifecycle hook

使用 [put-lifecycle-hook](#) 命令：

```
aws autoscaling put-lifecycle-hook \  
--lifecycle-hook-name my-termination-hook \  
--auto-scaling-group-name my-asg \  
--lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING \  
--default-result ABANDON \  
--heartbeat-timeout 300
```

管理保留的執行個體

監控並控制已移至保留狀態的 Amazon EC2 執行個體。使用 CloudWatch 指標追蹤保留的執行個體，然後在完成自訂動作後手動終止保留的執行個體。

保留的執行個體不會計入 Amazon EC2 Auto Scaling 群組所需的容量。當執行個體進入保留狀態時，Auto Scaling 會啟動替換執行個體以維持所需的容量。例如，假設 Auto Scaling 群組的所需容量為 10。當執行個體進入 Terminating:Retained 狀態時，Auto Scaling 會啟動替換執行個體，將所需的容量維持在 10。您現在總共有 11 個執行中的執行個體：作用中群組中有 10 個，加上 1 個保留的執行個體。所有 11 個執行個體的標準 Amazon EC2 費用將適用，直到您手動終止保留的執行個體為止。

保留執行個體的生命週期狀態

了解使用執行個體生命週期政策時，執行個體如何轉換生命週期狀態。執行個體遵循從正常終止到保留到最終終止的特定路徑。

觸發保留時，執行個體會轉換這些狀態：

1. Terminating - 正常終止開始
2. Terminating:Wait - 生命週期掛鉤執行
3. Terminating:Proceed - 生命週期動作後續處理（無論是成功還是失敗）
4. Terminating:Retained - 勾點失敗，執行個體保留以進行手動介入

暖集區執行個體會根據情況採取不同的生命週期狀態路徑：

執行個體擴展回暖集區：

1. `Warmed:Pending` - 開始正常暖集區轉換
2. `Warmed:Pending:Wait` - 生命週期掛鉤執行
3. `Warmed:Pending:Proceed` - 生命週期動作後續處理（無論是成功還是失敗）
4. `Warmed:Pending:Retained` - 勾點失敗，執行個體保留以進行手動介入

從暖集區終止的執行個體：

1. `Warmed:Terminating` - 正常終止開始
2. `Warmed:Terminating:Wait` - 生命週期掛鉤執行
3. `Warmed:Terminating:Proceed` - 生命週期動作後續處理（無論是否成功）
4. `Warmed:Terminating:Retained` - 勾點失敗，執行個體保留以進行手動介入

監控保留的執行個體

由於保留的 Amazon EC2 執行個體會產生成本並需要手動介入，因此監控它們至關重要。Amazon EC2 Auto Scaling 提供數個 CloudWatch 指標來追蹤保留的執行個體。

啟用群組指標以追蹤保留的執行個體：

```
aws autoscaling enable-metrics-collection \  
--auto-scaling-group-name my-asg \  
--metrics GroupTerminatingRetainedInstances
```

可用的指標為：

- `GroupTerminatingRetainedInstances` 顯示 `Terminating:Retained` 狀態中的執行個體數目。
- `GroupTerminatingRetainedCapacity` 顯示 `Terminating:Retained` 狀態的執行個體所代表的容量單位。
- `WarmPoolTerminatingRetainedCapacity` 會追蹤從暖集區終止的保留執行個體。
- `WarmPoolPendingRetainedCapacity` 會追蹤返回暖集區的保留執行個體。

您也可以檢查 Amazon EC2 Auto Scaling 群組的擴展活動，以了解保留執行個體的原因。尋找具有 `StatusCode: Cancelled` 和狀態原因訊息的終止活動，指出生命週期掛鉤失敗：

```
aws autoscaling describe-scaling-activities \  

```

```
--auto-scaling-group-name my-asg
```

我們建議您在這些指標上建立 CloudWatch 警示，以便在執行個體進入保留狀態時提醒您。這可協助您追蹤成本影響，並確保您不會忘記清除需要手動介入的執行個體。

終止保留的執行個體

完成自訂動作後，請呼叫 [TerminateInstanceInAutoScalingGroup](#) API 來終止保留的執行個體：

```
aws autoscaling terminate-instance-in-auto-scaling-group \  
--instance-id i-1234567890abcdef0 \  
--no-should-decrement-desired-capacity
```

透過執行個體中繼資料擷取目標生命週期狀態

您啟動的每個 Auto Scaling 執行個體都會經歷多個生命週期狀態。若要調用執行個體中的自訂動作，以便對特定的生命週期狀態轉換採取行動，您必須透過執行個體中繼資料擷取目標生命週期狀態。

例如，您可能需要一種機制來從執行個體內部偵測執行個體終止，以便在執行個體終止之前在其上執行某些程式碼。您可以撰寫程式碼，直接從執行個體輪詢執行個體的生命週期狀態來完成此操作。然後，您可以在 Auto Scaling 群組新增 lifecycle hook，讓執行個體保持運作，直到您的程式碼傳送 complete-lifecycle-action 命令以繼續操作。

Auto Scaling 執行個體生命週期具有兩個主穩定狀態 InService 和 Terminated，以及兩個副穩定狀態 Detached 和 Standby。如果使用暖集區，則生命週期還有四個額外的穩定狀態：Warmed:Hibernated、Warmed:Running、Warmed:Stopped 以及 Warmed:Terminated。

當執行個體準備轉移至上述穩定狀態之一時，Amazon EC2 Auto Scaling 會更新執行個體中繼資料項目 autoscaling/target-lifecycle-state 的值。若要從執行個體內獲取目標生命週期狀態，您必須使用執行個體中繼資料服務從執行個體中繼資料中進行擷取。

Note

執行個體中繼資料是有關 Amazon EC2 執行個體的資料，可供應用程式用來查詢執行個體資訊。執行個體中繼資料服務是一種執行個體上的元件，可供本機程式碼用來存取執行個體中繼資料。本機程式碼可以包括在執行個體上執行的使用者資料指令碼或應用程式。

本機程式碼可使用下列兩種方法之一，從執行中執行個體存取執行個體中繼資料：執行個體中繼資料服務第 1 版 (IMDSv1) 或執行個體中繼資料服務第 2 版 (IMDSv2)。IMDSv2 會使用工作階段導向的

請求，並減緩可能用來嘗試存取執行個體中繼資料的幾種漏洞。如需這兩種方法的詳細資訊，請參閱《Amazon EC2 [IMDSv2](#)》。Amazon EC2

IMDSv2

```
[ec2-user ~]$ TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` \
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

IMDSv1

```
[ec2-user ~]$ curl http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

以下為範例輸出。

```
InService
```

目標生命週期狀態是執行個體正要轉移到的狀態。目前的生命週期狀態是執行個體現在所處的狀態。在生命週期動作完成並且執行個體完成向目標生命週期狀態的轉移後，這二者可能相同。您無法從執行個體中繼資料中擷取執行個體目前的生命週期狀態。

Amazon EC2 Auto Scaling 於 2022 年 3 月 10 日開始生成目標生命週期狀態。如果您的執行個體在該日期之後轉換至目標生命週期狀態之一，則執行個體中繼資料中會出現目標生命週期狀態項目。否則，便會不出現，並且您會收到 HTTP 404 錯誤。

如需擷取執行個體中繼資料的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[擷取執行個體中繼資料](#)。

有關介紹如何在使用目標生命週期狀態的使用者資料指令碼中使用自訂動作建立 lifecycle hook 的教學課程，請參閱 [教學課程：使用資料指令碼和執行個體中繼資料來擷取生命週期狀態](#)。

Important

為了確保您可以盡快調用自訂動作，您的本機程式碼應該經常輪詢 IMDS，並在發生錯誤時重試。

將生命週期掛鉤新增至 Auto Scaling 群組

若要將 Auto Scaling 執行個體置於等待狀態並對其執行自訂動作，您可以將 lifecycle hook 新增至 Auto Scaling 群組。自訂動作會在執行個體啟動時或終止之前執行。在您完成生命週期動作前，或逾時期限結束前，執行個體都保持在等待狀態。

從建立 Auto Scaling 群組之後 AWS 管理主控台，您可以將一或多個生命週期掛鉤新增至其中，總共 50 個生命週期掛鉤。您也可以在建​​立 Auto Scaling 群組時 AWS CLI CloudFormation，使用 或 SDK 將生命週期掛鉤新增至該群組。

預設情況下，您在主控台中新增 lifecycle hook 時，Amazon EC2 Auto Scaling 便會向 Amazon EventBridge 傳送生命週期事件通知。使用 EventBridge 或使用者資料指令碼是建議的最佳實務。若要建立直接將通知傳送至 Amazon SNS、Amazon SQS 的生命週期關聯，或者 AWS Lambda 您可以使用 [put-lifecycle-hook](#) 命令，如本主題中的範例所示。

目錄

- [新增 lifecycle hook \(主控台\)](#)
- [新增 lifecycle hook \(AWS CLI\)](#)

新增 lifecycle hook (主控台)

請遵循以下步驟，將生命週期關聯新增至您的 Auto Scaling 群組。若要新增用於橫向擴展 (執行個體啟動) 和縮減 (執行個體終止或返回暖集區) 的生命週期關聯，您必須建立兩個單獨的關聯。

開始之前，請確認您已視需要設定自訂動作，如 [準備將 lifecycle hook 新增至您的 Auto Scaling 群組](#) 所述。

新增橫向擴展的 lifecycle hook

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。頁面底部會開啟一個分割窗格。
3. 在 Instance management (執行個體管理) 索引標籤的 Lifecycle hooks (lifecycle hook) 中，選擇 Create lifecycle hook (建立 lifecycle hook)。
4. 若要定義橫向擴展 (執行個體啟動) 的 lifecycle hook，請執行以下操作：
 - a. 對於 Lifecycle hook name (lifecycle hook 名稱)，指定 lifecycle hook 的名稱。

- b. 對於 Lifecycle transition (生命週期轉移)，選擇 Instance launch (執行個體啟動)。
- c. 在活動訊號逾時中，請指定在關聯逾時之前橫向擴展時，執行個體保持等待狀態的時間長度，單位為秒。範圍是從 30 至 7200 秒。設定較長的逾時期限，可為完成自訂動作提供更多時間。然後，如果您在逾時期限結束前完成，請傳送 [complete-lifecycle-action](#) 命令以允許執行個體繼續進入下一個狀態。
- d. 在 Default result (預設結果) 中，請指定當 lifecycle hook 逾時或發生意外失敗時要採取的動作。您可以選擇繼續或捨棄。
 - 如果您選擇繼續，Auto Scaling 群組可以用任何其他生命週期關聯繼續操作，然後將執行個體投入服務。
 - 如果選擇捨棄，Auto Scaling 群組會立即終止任何其餘的動作，並會立即終止執行個體。
- e. (選用) 針對通知中繼資料，指定在 Amazon EC2 Auto Scaling 將訊息傳送至通知目標時您想要包含的其他資訊。

5. 選擇建立。

新增縮減的 lifecycle hook

1. 選擇建立 lifecycle hook，即可在您建立橫向擴展 lifecycle hook 後的位置繼續處理。
2. 若要定義縮減的 lifecycle hook (執行個體終止或返回暖集區)，請執行以下操作：
 - a. 對於 Lifecycle hook name (lifecycle hook 名稱)，指定 lifecycle hook 的名稱。
 - b. 對於 Lifecycle transition (生命週期轉移)，選擇 Instance terminate (執行個體終止)。
 - c. 在活動訊號逾時中，請指定在關聯逾時之前橫向擴展時，執行個體保持等待狀態的時間長度，單位為秒。建議您使用 30 至 120 秒的短暫逾時時間，具體視您執行任何最終任務 (例如從 CloudWatch 提取 EC2 日誌) 所需的時間而定。
 - d. 對於 Default result (預設結果)，指定逾時或發生意外失敗時，Auto Scaling 群組會採取的動作。ABANDON (放棄) 和 CONTINUE (繼續) 皆允許執行個體終止。
 - 如果選擇 CONTINUE (繼續)，Auto Scaling 群組可以在終止前繼續執行任何剩餘的動作，如其他 lifecycle hook。
 - 如果選擇捨棄，Auto Scaling 群組會立即終止執行個體。
 - e. (選用) 針對通知中繼資料，指定在 Amazon EC2 Auto Scaling 將訊息傳送至通知目標時您想要包含的其他資訊。
3. 選擇建立。

新增 lifecycle hook (AWS CLI)

您可以使用 [put-lifecycle-hook](#) 命令建立和更新 lifecycle hook。

若要在擴增時執行某一動作，請使用下列命令。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

若要在縮減時執行某一動作，請使用下列命令。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING
```

若要使用 Amazon SNS 或 Amazon SQS 接收通知，請新增 `--notification-target-arn` 和 `--role-arn` 選項。若要使用接收通知 AWS Lambda，請新增 `--notification-target-arn`。

以下範例會建立了一個 lifecycle hook，該掛鉤會指定名為 *my-sns-topic* 的 SNS 主題作為通知目標。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING \  
  --notification-target-arn arn:aws:sns:region:123456789012:my-sns-topic \  
  --role-arn arn:aws:iam::123456789012:role/my-notification-role
```

該主題接收測試下列含有鍵值對的通知。

```
"Event": "autoscaling:TEST_NOTIFICATION"
```

在預設情況下，[put-lifecycle-hook](#) 命令會建立一個 lifecycle hook，其中包含 3600 秒 (1 小時) 的活動訊號逾時。

若要變更現有 lifecycle hook 的活動訊號逾時，請新增 `--heartbeat-timeout` 選項，如以下範例所示。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \  
  --auto-scaling-group-name my-asg --heartbeat-timeout 120
```


如果執行個體已處於等待狀態，您可以使用 [record-lifecycle-action-heartbeat](#) CLI 命令記錄活動訊號，以此防止 lifecycle hook 逾時。這樣做會將逾時期限延長到您在建立 lifecycle hook 時所指定的逾時值。如果在逾時期限結束前完成，您可以傳送 [complete-lifecycle-action](#) CLI 命令，以允許執行個體繼續進入下一個狀態。如需詳細資訊和範例，請參閱 [在 Auto Scaling 群組中完成生命週期動作](#)。

在 Auto Scaling 群組中完成生命週期動作

當 Auto Scaling 群組回應生命週期事件時，它會使執行個體進入等待狀態並傳送事件通知。當執行個體處於等待狀態，您可以執行自訂動作。

如果您在逾時期限到期之前完成生命週期動作且結果為 CONTINUE，將會對您有所幫助。如果您未完成生命週期動作，逾時期間結束後，lifecycle hook 便會移至您為預設結果指定的狀態。

目錄

- [完成生命週期動作 \(手動\)](#)
- [完成生命週期動作 \(自動\)](#)

完成生命週期動作 (手動)

下列步驟適用於命令列介面，不受主控台不支援。必須取代的資訊會以斜體顯示，例如執行個體 ID 或 Auto Scaling 群組的名稱。

完成生命週期動作 (AWS CLI)

1. 如果您需要更多時間來完成自訂動作，請使用 [record-lifecycle-action-heartbeat](#) 命令重新啟動逾時期限和保持執行個體的等待狀態。例如，如果逾時期限為 1 小時，且您在 30 分鐘後呼叫此命令，執行個體將再維持 1 小時的等待狀態，則總共為 90 分鐘。

您可以指定隨[通知](#)接收到的生命週期動作字符，如以下命令所示。

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg --lifecycle-action-  
token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

或者，您也可以指定您隨[通知](#)收到的執行個體 ID，如以下命令所示。

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \  
  --instance-id i-1234567890abcdef0
```

```
--auto-scaling-group-name my-asg --instance-id i-1a2b3c4d
```

2. 如果您在逾時期限結束之前完成自訂動作，請使用 [complete-lifecycle-action](#) 命令，讓 Auto Scaling 群組可以繼續啟動或終止執行個體。您可以指定生命週期動作字串，如以下命令所示。

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \  
--lifecycle-hook-name my-launch-hook --auto-scaling-group-name my-asg \  
--lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

或者，您也可以指定執行個體 ID，如下列命令所示。

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \  
--instance-id i-1a2b3c4d --lifecycle-hook-name my-launch-hook \  
--auto-scaling-group-name my-asg
```

完成生命週期動作 (自動)

如果擁有在啟動後設定執行個體的使用者資料指令碼，則您不需要手動完成生命週期動作。您可以新增 [complete-lifecycle-action](#) 命令至指令碼。該指令碼可以從執行個體中繼資料中擷取執行個體 ID，並在引導指令碼成功完成時向 Amazon EC2 Auto Scaling 傳送訊號。

如果您不是這樣做的，那麼更新您的指令碼來從執行個體的中繼資料擷取執行個體中的執行個體 ID。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[擷取執行個體中繼資料](#)。

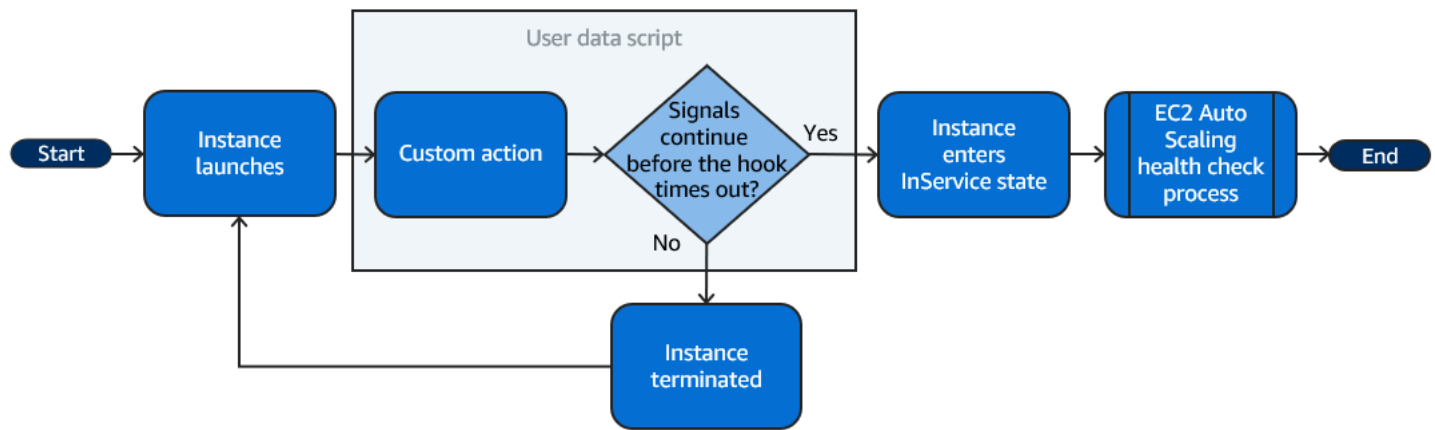
如果使用 Lambda，您也可以在此函數的程式碼中設定回呼，在自訂動作成功完成時，讓執行個體的生命週期繼續執行。如需詳細資訊，請參閱[教學課程：設定叫用 Lambda 函數的 lifecycle hook](#)。

教學課程：使用資料指令碼和執行個體中繼資料來擷取生命週期狀態

為 lifecycle hook 建立自訂動作的常見方法，是使用 Amazon EC2 Auto Scaling 傳送給其他服務 (如 Amazon EventBridge) 的通知。但是，您可以透過使用使用者資料指令碼將設定執行個體中繼資料並完成生命週期動作的程式碼移動至執行個體中繼資料本身，從而避免建立額外的基礎設施。

下列教學課程將介紹如何開始使用使用者資料指令碼和執行個體中繼資料。您可以使用使用者資料指令碼建立基本 Auto Scaling 群組配置，該指令碼可讀取群組中執行個體的[目標生命週期狀態](#)，並在執行個體生命週期的特定階段執行回呼動作以繼續啟動程序。

當您使用使用者資料指令碼執行自訂動作時，下圖摘要說明橫向擴展事件的流程。執行個體啟動後，透過逾時或 Amazon EC2 Auto Scaling 接收訊號以繼續，執行個體的生命週期會暫停，直到生命週期掛鉤完成為止。



目錄

- [步驟 1：建立具有完成生命週期動作所需許可的 IAM 角色](#)
- [步驟 2：建立啟動範本並加入 IAM 角色和使用者資料指令碼](#)
- [步驟 3：建立 Auto Scaling 群組](#)
- [步驟 4：新增 lifecycle hook](#)
- [步驟 5：測試並驗證功能](#)
- [步驟 6：清除](#)
- [相關資源](#)

步驟 1：建立具有完成生命週期動作所需許可的 IAM 角色

當您使用 AWS CLI 或 AWS 開發套件傳送回呼以完成生命週期動作時，您必須使用具有完成生命週期動作許可的 IAM 角色。

建立政策

1. 開啟 IAM 主控台的[政策頁面](#)，然後選擇 Create policy (建立政策)。
2. 選擇 JSON 標籤。
3. 在 Policy Document (政策文件) 方塊中，將下列政策文件複製並貼入方塊。將 *sample text* 取代之為您的帳戶號碼以及您要建立的 Auto Scaling 群組名稱 (**TestAutoScalingEvent-group**)。

JSON

```
{
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "autoscaling:CompleteLifecycleAction"
    ],
    "Resource":
      "arn:aws:autoscaling:*:123456789012:autoScalingGroup*:autoScalingGroupName/
      TestAutoScalingEvent-group"
  }
]
```

4. 選擇下一步。
5. 針對 Policy name (政策名稱)，輸入 **TestAutoScalingEvent-policy**。選擇建立政策。

當您完成建立政策時，您可以建立一個使用它的角色。

建立角色

1. 在左側導覽窗格中，選擇 Roles (角色)。
2. 選擇建立角色。
3. 對於 Select trusted entity (選取信任的實體) 區段，選擇 AWS service (AWS 服務)。
4. 對於您的使用案例，選擇 EC2，然後選擇 Next (下一步)。
5. 在 Add permissions (新增許可) 下，選擇您建立的政策 (TestAutoScalingEvent-policy)。然後選擇下一步。
6. 在 Name, review, and create (命名、檢閱和建立) 頁面上的 Role name (角色名稱)，輸入 **TestAutoScalingEvent-role**，然後選擇 Create role (建立角色)。

步驟 2：建立啟動範本並加入 IAM 角色和使用者資料指令碼

建立可搭配 Auto Scaling 群組使用的啟動範本。在其中加入您建立的 IAM 角色和提供的範例使用者資料指令碼。

建立啟動範本

1. 開啟 Amazon EC2 主控台中的[啟動範本頁面](#)。
2. 選擇 Create launch template (建立啟動範本)。

3. 針對 Launch template name (啟動範本名稱)，請輸入 **TestAutoScalingEvent-template**。
4. 在 Auto Scaling guidance (Auto Scaling 指引) 下，選取核取方塊。
5. 在 Application and OS Images (Amazon Machine Image) (應用程式和作業系統映像 (Amazon Machine Image)) 欄位中，請從 Quick Start (快速入門) 清單中選擇 Amazon Linux 2 (HVM)、SSD 磁碟區類型、64 位元 (x86)。
6. 在 Instance type (執行個體類型) 中，選擇 Amazon EC2 執行個體的類型 (例如 t2.micro)。
7. 針對 Advanced details (進階詳細資訊)，請展開此區段來檢視欄位。
8. 在 IAM instance profile (IAM 執行個體設定檔) 中，選擇 IAM 角色 (TestAutoScalingEvent-role) 的 IAM 執行個體設定檔名稱。執行個體設定檔是 IAM 角色的容器，而此角色允許 Amazon EC2 在執行個體啟動時將 IAM 角色傳遞至該執行個體。

當您使用 IAM 主控台建立 IAM 角色時，主控台會自動建立一個與其對應角色同名的執行個體設定檔。

9. 對於 User data (使用者資料)，請複製以下範例使用者資料指令碼並貼入該欄位。將 的範例文字取代 group_name 為您要建立的 Auto Scaling 群組名稱，並將 region 取代為 AWS 區域 您希望 Auto Scaling 群組使用的。

```
#!/bin/bash

function token {
    echo "X-aws-ec2-metadata-token: $(curl -X PUT 'http://169.254.169.254/latest/api/token' -H 'X-aws-ec2-metadata-token-ttl-seconds: 21600')"
}

function get_target_state {
    echo $(curl -H "$(token)" -s http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state)
}

function get_instance_id {
    echo $(curl -H "$(token)" -s http://169.254.169.254/latest/meta-data/instance-id)
}

function complete_lifecycle_action {
    instance_id=$(get_instance_id)
    group_name='TestAutoScalingEvent-group'
    region='us-west-2'
```

```
    echo $instance_id
    echo $region
    echo $(aws autoscaling complete-lifecycle-action \
        --lifecycle-hook-name TestAutoScalingEvent-hook \
        --auto-scaling-group-name $group_name \
        --lifecycle-action-result CONTINUE \
        --instance-id $instance_id \
        --region $region)
}

function main {
    while true
    do
        target_state=$(get_target_state)
        if [ \"$target_state\" = \"InService\" ]; then
            # Change hostname
            export new_hostname=\"${group_name}-${instance_id}\"
            hostname $new_hostname
            # Send callback
            complete_lifecycle_action
            break
        fi
        echo $target_state
        sleep 5
    done
}

main
```

此簡單的使用者資料指令碼會執行下列操作：

- 叫用執行個體中繼資料中繼資料，以從執行個體中繼資料中擷取目標生命週期狀態和執行個體 ID
- 重複擷取目標生命週期狀態，直到狀態變更為 InService
- 如果目標生命週期狀態為 InService，請將執行個體的主機名稱變更為帶有 Auto Scaling 群組名稱的執行個體 ID
- 透過叫用 complete-lifecycle-action CLI 命令傳送回呼，以便指示 Amazon EC2 Auto Scaling 繼續 (CONTINUE) EC2 啟動程序

10. 選擇 Create launch template (建立啟動範本)。

11. 在確認頁面上，選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

Note

如需其他可用作開發使用者資料指令碼參考的範例，請參閱 Amazon EC2 Auto Scaling 的 [GitHub 儲存庫](#)。

步驟 3：建立 Auto Scaling 群組

建立啟動範本後，請建立 Auto Scaling 群組。

建立 Auto Scaling 群組

1. 在 Choose launch template or configuration (選擇啟動範本或組態) 頁面上，請就 Auto Scaling group name (Auto Scaling 群組名稱) 輸入 Auto Scaling 群組的名稱 (**TestAutoScalingEvent-group**)。
2. 選擇 Next (下一步)，以前往 Choose instance launch options (選擇執行個體啟動選項) 頁面。
3. 在 Network (網路) 中，選擇 VPC。
4. 在 Availability Zones and subnets (可用區域和子網路) 中，請從一或多個可用區域中選擇一或多個子網路。
5. 在 Instance type requirements (執行個體類型需求) 區段中，請使用預設設定來簡化此步驟。(請勿覆寫啟動範本。) 在本教程中，您將僅用啟動範本中指定的執行個體類型僅啟動一個隨需執行個體。
6. 選擇畫面底部的 Skip to review (跳過以進行檢閱)。
7. 在 Review (檢閱) 頁面中，檢閱 Auto Scaling 群組的詳細資訊，然後選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。

步驟 4：新增 lifecycle hook

新增 lifecycle hook，使執行個體保持等待狀態，直到您的生命週期動作完成。

新增 lifecycle hook

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組旁的核取方塊。頁面底部會開啟一個分割窗格。
3. 在下方窗格中，在 Instance management (執行個體管理) 索引標籤的 Lifecycle hooks (lifecycle hook) 中，選擇 Create lifecycle hook (建立 lifecycle hook)。

4. 若要定義橫向擴展 (執行個體啟動) 的 lifecycle hook，請執行以下操作：
 - a. 對於 Lifecycle hook name (lifecycle hook 名稱)，輸入 **TestAutoScalingEvent-hook**。
 - b. 對於 Lifecycle transition (生命週期轉移)，選擇 Instance launch (執行個體啟動)。
 - c. 在 Heartbeat timeout (活動訊號逾時) 中輸入 **300**，表示等待使用者資料指令碼回呼的秒數。
 - d. 在 Default result (預設結果) 中，選擇 ABANDON (放棄)。如果掛鉤逾時而未收到來自您使用者資料指令碼的回呼，Auto Scaling 群組會終止新的執行個體。
 - e. (選用) 將 Notification metadata (通知中繼資料) 保留空白。
5. 選擇建立。

步驟 5：測試並驗證功能

若要測試功能，請將 Auto Scaling 群組的容量增加 1，以便更新 Auto Scaling 群組。在執行個體啟動後不久，使用者資料指令碼就會執行並開始檢查執行個體的目標生命週期狀態。當目標生命週期狀態為 InService 時，指令碼會變更主機名稱並傳送回呼動作。完成此過程通常只需要幾秒鐘的時間。

增加 Auto Scaling 群組的大小

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組旁的核取方塊。在下方窗格中檢視詳細資料時，仍可看到上方窗格的頂端列。
3. 在下方窗格中，在 Details (詳細資訊) 索引標籤上選擇 Group details (群組詳細資訊)、Edit (編輯)。
4. 對於 Desired capacity (所需容量)，將目前值增加 1。
5. 選擇更新。正在啟動執行個體時，上方窗格中的 Status (狀態) 欄會顯示 Updating capacity (更新容量) 狀態。

增加所需容量後，您可以確認執行個體已成功啟動，並且未從擴展活動的描述中終止。

檢視擴展活動

1. 返回 Auto Scaling groups (Auto Scaling 群組) 頁面並選取群組。
2. 在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 下方，Status (狀態) 欄會顯示 Auto Scaling 群組是否已成功啟動執行個體。
3. 如果使用者資料指令碼失敗，則在逾時期限過後，您會看到狀態為 Canceled 的擴展活動，以及狀態消息 Instance failed to complete user's Lifecycle Action: Lifecycle

Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned:
Lifecycle Action Completed with ABANDON Result。

步驟 6：清除

如果您已完成使用針對此教學課程建立的資源，請按下列步驟將其刪除。

刪除 lifecycle hook

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組旁的核取方塊。
3. 在 Instance management (執行個體管理) 索引標籤的 Lifecycle hooks (lifecycle hook) 中，選擇 lifecycle hook (TestAutoScalingEvent-hook)。
4. 選擇 動作、刪除。
5. 再次選擇 Delete (刪除) 進行確認。

刪除啟動範本

1. 開啟 Amazon EC2 主控台內的 [啟動範本頁面](#)。
2. 選取啟動範本 (TestAutoScalingEvent-template)，然後選擇 Actions (動作)、Delete template (刪除範本)。
3. 出現確認提示時，請輸入 **Delete** 確認刪除指定的啟動範本，然後選擇 Delete (刪除)。

若已完成使用範例 Auto Scaling 群組，請將其刪除。您也可以刪除您建立的 IAM 角色和許可政策。

刪除 Auto Scaling 群組

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組旁邊的核取方塊 (TestAutoScalingEvent-group)，然後選擇 Delete (刪除)。
3. 出現確認提示時，請輸入 **delete** 來確認刪除特定的 Auto Scaling 群組，然後選擇 Delete (刪除)。

Name (名稱) 欄位中的載入圖示會顯示正在刪除 Auto Scaling 群組。終止執行個體並刪除群組需要幾分鐘的時間。

刪除 IAM 角色

1. 開啟 IAM 主控台中的 [Roles](#) (角色) 頁面。
2. 選取函數的角色 (TestAutoScalingEvent-role)。
3. 選擇 刪除。
4. 出現確認提示時，請輸入角色名稱，然後選擇 Delete (刪除)。

刪除 IAM 政策

1. 開啟 IAM 主控台中的 [政策頁面](#)。
2. 選取您建立的政策 (TestAutoScalingEvent-policy)。
3. 選擇 動作、刪除。
4. 出現確認提示時，請輸入政策名稱，然後選擇 Delete (刪除)。

相關資源

當您根據執行個體中繼資料中提供的資料，開發針對執行個體調用動作的程式碼時，下列相關主題將對您有所幫助。

- [透過執行個體中繼資料擷取目標生命週期狀態](#). 本節說明其他使用案例 (例如執行個體終止) 的生命週期狀態。
- [新增 lifecycle hook \(主控台\)](#). 此程序說明如何為橫向擴展 (執行個體啟動) 和縮減 (執行個體終止或返回暖集區) 新增生命週期關聯。
- 《Amazon EC2 使用者指南》中的 [執行個體中繼資料類別](#)。本主題列出可用於在 EC2 執行個體上叫用動作的所有執行個體中繼資料類別。

如需說明如何使用 Amazon EventBridge 建立規則，以根據 Auto Scaling 群組中執行個體發生的事件調用 Lambda 函數的教學課程，請參閱 [教學課程：設定叫用 Lambda 函數的 lifecycle hook](#)。

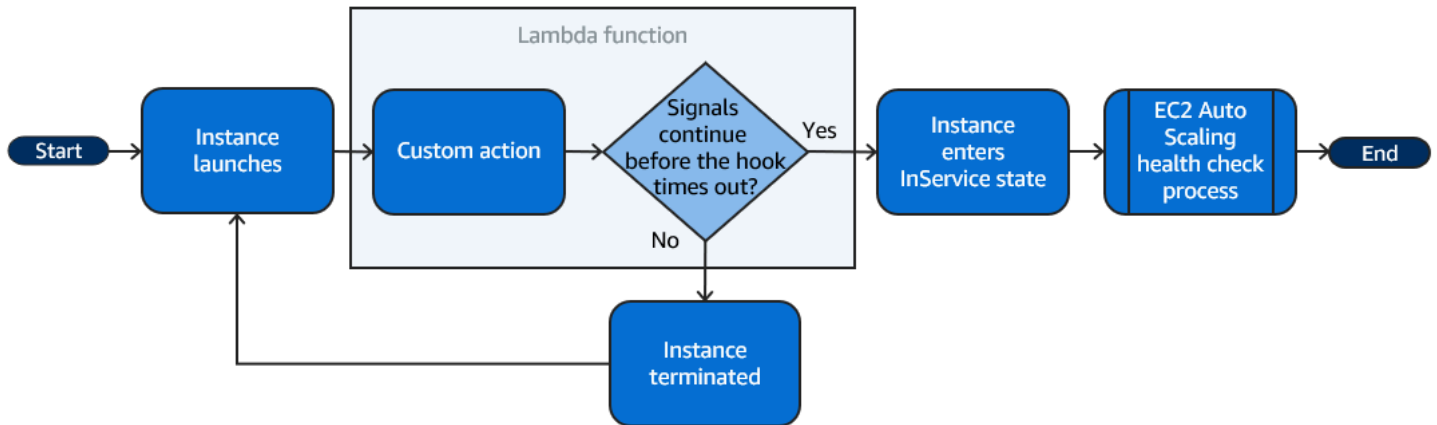
教學課程：設定叫用 Lambda 函數的 lifecycle hook

在本練習中，您會建立 Amazon EventBridge 規則，其中包含篩選條件模式，在相符時呼叫 AWS Lambda 函數做為規則目標。我們提供要使用的篩選模式和範例函數程式碼。

如果一切設定正確，則在此教學課程的結尾，Lambda 函數會在執行個體啟動時執行自訂動作。自訂動作只會將事件記錄在與 Lambda 函數相關聯的 CloudWatch Logs 日誌串流中。

Lambda 函數也會執行回呼，讓執行個體的生命週期在此動作成功時繼續執行，但在動作失敗時讓執行個體放棄啟動並終止。

當您使用 Lambda 函數執行自訂動作時，下圖摘要說明橫向擴展事件的流程。執行個體啟動後，透過逾時或 Amazon EC2 Auto Scaling 接收訊號以繼續，執行個體的生命週期會暫停，直到生命週期掛鉤完成為止。



Note

根據您的使用案例，您可以依照下列步驟並建立 EventBridge 規則來設定生命週期掛鉤。或者，您可以使用 Lambda 函數直接設定生命週期關聯，而無需建立 EventBridge 規則。

目錄

- [先決條件](#)
- [步驟 1：建立具有完成生命週期動作所需許可的 IAM 角色](#)
- [步驟 2：建立 Lambda 函數](#)
- [步驟 3：建立 EventBridge 規則](#)
- [步驟 4：新增生命週期掛鉤](#)
- [步驟 5：測試並驗證事件](#)
- [步驟 6：清除](#)
- [相關資源](#)

先決條件

開始此教學課程前，請先建立 Auto Scaling 群組 (如果尚未建立)。若要建立 Auto Scaling 群組，請開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)，再選擇建立 Auto Scaling 群組。

步驟 1：建立具有完成生命週期動作所需許可的 IAM 角色

建立 Lambda 函數之前，您必須先建立執行角色和許可政策，才能允許 Lambda 完成 lifecycle hook。

建立政策

1. 開啟 IAM 主控台的[政策頁面](#)，然後選擇 Create policy (建立政策)。
2. 選擇 JSON 標籤。
3. 在 Policy Document (政策文件) 方塊中，將下列政策文件貼入方塊，取代##顯示文字，並提供您的帳戶號碼和 Auto Scaling 群組的名稱。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
        "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/my-asg"
    }
  ]
}
```

4. 選擇下一步。
5. 針對 Policy name (政策名稱)，輸入 **LogAutoScalingEvent-policy**。選擇建立政策。

當您完成建立政策時，您可以建立一個使用它的角色。

建立角色

1. 在左側導覽窗格中，選擇 Roles (角色)。
2. 選擇建立角色。
3. 對於 Select trusted entity (選取信任的實體) 區段，選擇 AWS service (AWS 服務)。
4. 對於您的使用案例，選擇 Lambda，然後選擇 Next (下一步)。
5. 在 Add permissions (新增許可) 下，選擇您建立的政策 (LogAutoScalingEvent-policy) 和名為 AWSLambdaBasicExecutionRole 的政策。然後選擇下一步。

Note

AWSLambdaBasicExecutionRole 政策具備函數將日誌寫入到 CloudWatch Logs 時所需的許可。

6. 在 Name, review, and create (命名、檢閱和建立) 頁面上的 Role name (角色名稱)，輸入 **LogAutoScalingEvent-role**，然後選擇 Create role (建立角色)。

步驟 2：建立 Lambda 函數

建立 Lambda 函數作為事件的目標。在 Amazon EC2 Auto Scaling 發出相符事件時，EventBridge 會叫用以 Node.js 撰寫的範例 Lambda 函數。

建立 Lambda 函數

1. 開啟 Lambda 主控台中的 [Functions \(函數\) 頁面](#)。
2. 依序選擇 Create function (建立函數)、Author from scratch (從頭開始撰寫)。
3. 在 Basic information (基本資訊) 下，為 Function name (函數名稱) 輸入 **LogAutoScalingEvent**。
4. 在執行期選擇 Node.js 18.x。
5. 向下捲動並選擇變更預設執行角色，然後在執行角色中選擇使用現有角色。
6. 在 Existing role (現有角色) 中，選擇 LogAutoScalingEvent-role。
7. 保留其他預設值。
8. 選擇 Create function (建立函數)。您會回到該函數的程式碼和組態畫面。
9. 在主控台中保持 LogAutoScalingEvent 函數開啟，在編輯器中的程式碼來源下，將下列範本程式碼貼到名為 index.js 的檔案中。

```
import { AutoScalingClient, CompleteLifecycleActionCommand } from "@aws-sdk/client-
auto-scaling";
export const handler = async(event) => {
  console.log('LogAutoScalingEvent');
  console.log('Received event:', JSON.stringify(event, null, 2));
  var autoscaling = new AutoScalingClient({ region: event.region });
  var eventDetail = event.detail;
  var params = {
    AutoScalingGroupName: eventDetail['AutoScalingGroupName'], /* required */
    LifecycleActionResult: 'CONTINUE', /* required */
    LifecycleHookName: eventDetail['LifecycleHookName'], /* required */
    InstanceId: eventDetail['EC2InstanceId'],
    LifecycleActionToken: eventDetail['LifecycleActionToken']
  };
  var response;
  const command = new CompleteLifecycleActionCommand(params);
  try {
    var data = await autoscaling.send(command);
    console.log(data); // successful response
    response = {
      statusCode: 200,
      body: JSON.stringify('SUCCESS'),
    };
  } catch (err) {
    console.log(err, err.stack); // an error occurred
    response = {
      statusCode: 500,
      body: JSON.stringify('ERROR'),
    };
  }
  return response;
};
```

此程式碼會直接記錄事件，以便您能在此教學課程的最後看到 CloudWatch Logs 日誌串流中出現與此 Lambda 函數相關聯的事件。

10. 選擇 Deploy (部署)。

步驟 3：建立 EventBridge 規則

建立 EventBridge 規則來執行 Lambda 函數。如需有關使用 EventBridge 的詳細資訊，請參閱 [使用 EventBridge 處理 Auto Scaling 事件](#)。

使用主控台建立規則

1. 開啟 [EventBridge 主控台](#)。
2. 在導覽窗格中，選擇規則。
3. 選擇建立規則。
4. 針對 Define rule detail (定義規則詳細資訊) 執行下列動作：
 - a. 對於名稱，輸入 **LogAutoScalingEvent-rule**。
 - b. 針對事件匯流排選擇預設值。當您 AWS 服務 帳戶中的 產生事件時，一律會前往您帳戶的預設事件匯流排。
 - c. 針對規則類型，選擇具有事件模式的規則。
 - d. 選擇下一步。
5. 針對 Build event pattern (建置事件模式) 執行下列動作：
 - a. 在 Event source (事件來源) 欄位中，選擇 AWS events or EventBridge partner events (事件或 EventBridge 合作夥伴事件)。
 - b. 向下捲動至事件模式，然後執行以下操作：
 - c.
 - i. 在 Event source (事件來源)，選擇 AWS 服務。
 - ii. 對於 AWS 服務，選擇 Auto Scaling。
 - iii. 在 Event Type (事件類型) 中，選擇 Instance Launch and Terminate (執行個體啟動和終止)。
 - iv. 在預設情況下，規則會符合任何縮減或橫向擴展事件。若要建立規則，在出現橫向擴展事件且執行個體因 lifecycle hook 而進入等待狀態時通知您，請選擇 Specific instance event(s) (特定執行個體事件) 並選取 EC2 Instance-launch Lifecycle Action (EC2 執行個體啟動生命週期動作)。
 - v. 在預設情況下，規則符合區域中的任何 Auto Scaling 群組。為了讓規則符合特定 Auto Scaling 群組，請選擇特定群組名稱，然後選取一個群組。
 - vi. 選擇下一步。
6. 針對 Select target(s) (選取目標) 執行下列動作：

- a. 對於 Target types (目標類型)，選擇 AWS 服務。
 - b. 對於 Select a target (選取目標)，選擇 Lambda function (Lambda 函數)。
 - c. 對於 Function (函數)，選擇 LogAutoScalingEvent。
 - d. 選擇 Next (下一步) 兩次。
7. 在檢閱和建立頁面上，選取建立規則。

步驟 4：新增生命週期掛鉤

在本節中，您會新增 lifecycle hook，以便 Lambda 在啟動時在執行個體上執行函數。

新增 lifecycle hook

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組旁的核取方塊。頁面底部會開啟一個分割窗格。
3. 在下方窗格中，在 Instance management (執行個體管理) 索引標籤的 Lifecycle hooks (lifecycle hook) 中，選擇 Create lifecycle hook (建立 lifecycle hook)。
4. 若要定義橫向擴展 (執行個體啟動) 的 lifecycle hook，請執行以下操作：
 - a. 對於 Lifecycle hook name (lifecycle hook 名稱)，輸入 **LogAutoScalingEvent-hook**。
 - b. 對於 Lifecycle transition (生命週期轉移)，選擇 Instance launch (執行個體啟動)。
 - c. 對於 Heartbeat timeout (活動訊號逾時)，輸入 **300**，表示等待 Lambda 函數回呼的秒數。
 - d. 在 Default result (預設結果) 中，選擇 ABANDON (放棄)。這表示如果掛鉤逾時而未收到來自 Lambda 函數的回呼，Auto Scaling 群組會終止新的執行個體。
 - e. (選用) 讓 Notification metadata (通知中繼資料) 保持空白。我們傳遞給 EventBridge 的事件資料包含所有叫用 Lambda 函數的必要資訊。
5. 選擇建立。

步驟 5：測試並驗證事件

若要測試事件，請將 Auto Scaling 群組的容量增加 1，以便更新 Auto Scaling 群組。系統會在增加所需容量後幾秒內叫用 Lambda 函數。

增加 Auto Scaling 群組的大小

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。

2. 選取 Auto Scaling 群組旁邊的核取方塊，即可在下方窗格中檢視詳細資訊，並仍可查看上方窗格的頂端列。
3. 在下方窗格中，在 Details (詳細資訊) 索引標籤上選擇 Group details (群組詳細資訊)、Edit (編輯)。
4. 對於 Desired capacity (所需容量)，將目前值增加 1。
5. 選擇更新。正在啟動執行個體時，上方窗格中的 Status (狀態) 欄會顯示 Updating capacity (更新容量) 狀態。

增加所需容量後，您可以確認 Lambda 函數是否被叫用。

檢視 Lambda 函數的輸出

1. 開啟 CloudWatch 主控台的 [日誌群組](#) 頁面。
2. 為 Lambda 函數 (/aws/lambda/LogAutoScalingEvent) 選取日誌群組名稱。
3. 選取日誌串流的名稱，以便檢視函數為生命週期動作所提供的資料。

接下來，您可以從擴展活動的描述中確認執行個體是否成功啟動。

檢視擴展活動

1. 返回 Auto Scaling groups (Auto Scaling 群組) 頁面並選取群組。
2. 在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 下方，Status (狀態) 欄會顯示 Auto Scaling 群組是否已成功啟動執行個體。
 - 如果動作成功，擴展活動的狀態將為 Successful (成功)。
 - 如果失敗，在等待幾分鐘之後，您會看到狀態為 Cancelled (已取消) 的擴展活動，以及這樣的狀態訊息：Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result (執行個體無法完成使用者的生命週期動作：字符為 e85eb647-4fe0-4909-b341-a6c42EXAMPLE 的生命週期動作已遭放棄：生命週期動作完成，結果為放棄)。

降低 Auto Scaling 群組的大小

如果不需要在此測試中啟動的多餘執行個體，可以開啟 Details (詳細資訊) 索引標籤，將 Desired capacity (所需容量) 減少 1。

步驟 6：清除

如果您已完成使用只針對此教學課程建立的資源，請按下列步驟將其刪除。

刪除生命週期掛鉤

1. 開啟 Amazon EC2 主控台的 [Auto Scaling 群組頁面](#)。
2. 選取 Auto Scaling 群組旁的核取方塊。
3. 在 Instance management (執行個體管理) 索引標籤的 Lifecycle hooks (lifecycle hook) 中，選擇 lifecycle hook (LogAutoScalingEvent-hook)。
4. 選擇 動作、刪除。
5. 再次選擇 Delete (刪除) 進行確認。

刪除 Amazon EventBridge 規則

1. 在 Amazon EventBridge 主控台中開啟 [Rules \(規則\) 頁面](#)。
2. 在 Event bus (事件匯流排) 下，選擇與規則相關聯的事件匯流排 (Default)。
3. 選取規則 (LogAutoScalingEvent-rule) 旁的核取方塊。
4. 選擇 刪除。
5. 出現確認提示時，請輸入規則名稱，然後選擇 Delete (刪除)。

若已完成使用範例函數，請將其刪除。您還可以刪除存放函數日誌的日誌群組，以及您建立的執行角色和許可政策。

刪除 Lambda 函數

1. 開啟 Lambda 主控台中的 [Functions \(函數\) 頁面](#)。
2. 選擇函數 (LogAutoScalingEvent)。
3. 選擇 動作、刪除。
4. 出現確認提示時，請輸入 **delete** 確認刪除指定的函數，然後選擇 Delete (刪除)。

刪除日誌群組

1. 開啟 CloudWatch 主控台的 [日誌群組](#) 頁面。
2. 選取函數的日誌群組 (/aws/lambda/LogAutoScalingEvent)。

3. 選擇 動作、刪除日誌群組。
4. 在 刪除日誌群組 對話方塊中，選擇 刪除。

刪除執行角色

1. 開啟 IAM 主控台中的 [Roles](#) (角色) 頁面。
2. 選取函數的角色 (LogAutoScalingEvent-role)。
3. 選擇 刪除。
4. 出現確認提示時，請輸入角色名稱，然後選擇 Delete (刪除)。

刪除 IAM 政策

1. 開啟 IAM 主控台中的 [政策頁面](#)。
2. 選取您建立的政策 (LogAutoScalingEvent-policy)。
3. 選擇 動作、刪除。
4. 出現確認提示時，請輸入政策名稱，然後選擇 Delete (刪除)。

相關資源

當您根據 Auto Scaling 群組中執行個體發生的事件建立 EventBridge 規則時，下列相關主題可能會有所幫助。

- [使用 EventBridge 處理 Auto Scaling 事件](#). 本節列出其他使用案例的事件範例，包括縮減事件。
- [新增 lifecycle hook \(主控台\)](#). 此程序說明如何為橫向擴展 (執行個體啟動) 和縮減 (執行個體終止或返回暖集區) 新增生命週期關聯。

如需說明如何使用執行個體中繼資料服務 (IMDS) 從執行個體本身調用動作的教學課程，請參閱 [教學課程：使用資料指令碼和執行個體中繼資料來擷取生命週期狀態](#)。

使用暖集區縮短開機時間的應用程式延遲

暖集區可讓您將開機時間非常長的應用程式 (例如因為執行個體需要將大量資料寫入磁碟) 的延遲。使用暖集區讓您不再需要過度佈建 Auto Scaling 群組來管理延遲，以改善應用程式效能。如需詳細資訊，請參閱以下部落格文章：[使用 EC2 Auto Scaling 暖集區更快擴展應用程式](#)。

Important

若在不需要的情況下建立暖集區，可能會導致不必要的成本。如果首次開機時間不會對應用程式造成明顯的延遲問題，則可能不需要使用暖集區。

主題

- [核心概念](#)
- [先決條件](#)
- [更新暖集區中的執行個體](#)
- [相關資源](#)
- [限制](#)
- [在 Auto Scaling 群組中使用具有暖集區的生命週期掛鉤](#)
- [為 Auto Scaling 群組建立暖集區](#)
- [檢視運作狀態檢查狀態和運作狀態檢查失敗的原因](#)
- [使用 建立和管理暖集區的範例 AWS CLI](#)

核心概念

在開始使用之前，請熟悉以下核心概念：

暖集區

暖集區是位於 Auto Scaling 群組旁的預先初始化 EC2 執行個體集區。每當應用程式需要擴增時，Auto Scaling 群組都可以利用暖集區來符合其所需的新容量。這有助於您確保執行個體已準備好快速開始為應用程式流量提供服務，加速對水平擴展事件的回應。在執行個體離開暖集區時，會計入群組所需的容量。這稱為暖啟動。

執行個體在暖集區時，只有在處於 InService 狀態執行個體的指標值大於擴展政策的高警示閾值（其與目標追蹤擴展政策的目標使用率相同）時，您的擴展政策才會水平擴展。

暖集區大小

根據預設，暖集區大小計算方式為 Auto Scaling 群組的最大容量及其所需容量之間的差異。例如，如果 Auto Scaling 群組的所需容量為 6，且最大容量為 10，則在您第一次設定暖集區且集區正在初始化時，暖集區的大小將為 4。

若要個別指定暖集區的最大容量，請使用自訂規格 (MaxGroupPreparedCapacity) 選項，並為其設定大於 群組目前容量的自訂值。如果您提供自訂值，暖集區的大小會計算為自訂值與群組目前所需容量之間的差異。例如，如果 Auto Scaling 群組的所需容量為 6，如果最大容量為 20，且自訂值為 8，則當您第一次設定暖集區且集區正在初始化時，暖集區的大小將為 2。

使用大型 Auto Scaling 群組時，您可能只需要使用自訂規格 (MaxGroupPreparedCapacity) 選項，即可管理擁有暖集區的成本優勢。例如，某個 Auto Scaling 群組具有 1,000 個執行個體且容量上限為 1,500 (以提供處理緊急流量尖峰的額外容量)，包含 100 個執行個體的暖集區，比起保存 500 個可供未來在暖集區內使用的預留執行個體，可能對於達成目標更有幫助。

Minimum warm pool size (暖集區大小下限)

考慮使用最小大小設定 (MinSize) 來靜態設定要在暖集區中維護的執行個體數量下限。預設為沒有設定大小下限。當您指定 MaxGroupPreparedCapacity 以確保即使在 Auto Scaling 群組的所需容量高於 時，在暖集區中維持最少數量的執行個體時，此 MinSize 設定很有用 MaxGroupPreparedCapacity。

暖集區執行個體狀態

您可以將暖集區中的執行個體維持在以下三種狀態之一：Stopped、Running 或 Hibernated。保持執行個體為 Stopped 狀態是一種盡量減少成本的有效方式。停止執行個體後，您只需為使用的磁碟區和連接至執行個體的彈性 IP 地址付費。

或者，您也可以將執行個體維持在 Hibernated 狀態，即可停止執行個體而不刪除其記憶體內容 (RAM)。若執行個體處於休眠狀態，就會向作業系統傳送訊號，將 RAM 的內容儲存至 Amazon EBS 根磁碟區中。當執行個體再次啟動時，根磁碟區會還原至其先前的狀態，並重新載入 RAM 內容。若執行個體處於休眠狀態，您就只需為 EBS 磁碟區付費，包括 RAM 內容的儲存，以及連接至執行個體的彈性 IP 地址。

也可以將暖集區中的執行個體維持在 Running 狀態，但不建議這樣做，以避免產生不必要的費用。當執行個體停止或休眠時，您可以節省執行個體本身的成本。只有在執行個體執行時，才需支付它們的費用。

lifecycle hook

您使用 [生命週期關聯](#) 將執行個體置於等待狀態，以便您對執行個體執行自訂動作。自訂動作會在執行個體啟動時或終止之前執行。

在暖集區組態中，生命週期關聯可以延遲執行個體停止或休眠，以及延遲在橫向擴展事件期間投入使用，直到其完成初始化。如果在沒有 lifecycle hook 的情況下向 Auto Scaling 群組新增暖集區，需要很長時間才能完成初始化的執行個體可能會停止或休眠，接著在準備就緒前，即在水平擴展事件期間投入使用。

執行個體重複使用政策

依預設，當 Auto Scaling 群組縮減時，Amazon EC2 Auto Scaling 會終止您的執行個體。然後，Amazon EC2 Auto Scaling 會將新執行個體啟動到暖集區內，取代已終止的執行個體。

如果您希望將執行個體傳回暖集區，可以指定執行個體重複使用政策。如此一來，您就可以重複使用已設定為應用程式流量提供服務的執行個體。為確保您的暖集區不會過度佈建，Amazon EC2 Auto Scaling 可以根據其設定終止暖集區中的執行個體，以便在暖集區大於所需大小時將其縮小。終止暖集區中的執行個體時，會使用[預設終止政策](#)選擇要首先終止哪些執行個體。

Important

如果您想要在縮減時使執行個體休眠，且 Auto Scaling 群組中存在現有執行個體，則這些執行個體必須滿足執行個體休眠的要求。如果不符合要求，執行個體返回暖集區時，便會回退到停止狀態，而不是休眠狀態。

Note

目前，您只能使用 AWS CLI 或 SDK 來指定執行個體重複使用政策。這項功能在主控台中無法使用。

先決條件

為 Auto Scaling 群組建立暖集區之前，請先決定如何使用生命週期關聯以適當的初始狀態來初始化新執行個體。

若要在執行個體因 lifecycle hook 而處於等待狀態時對執行個體執行自訂動作，您有兩個選項：

- 針對要在啟動時在執行個體上執行命令的簡單案例，您可以在為 Auto Scaling 群組建立啟動範本或啟動組態時，納入使用者資料指令碼。使用者資料指令碼只是執行個體啟動cloud-init時由執行的正常 shell 指令碼或cloud-init指令。指令碼還可以使用執行此指令碼的執行個體的 ID，藉此控制執行個體何時轉換到下一個狀態。如果您不是這樣做的，那麼更新您的指令碼來從執行個體的中繼資料擷取執行個體中的執行個體 ID。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[存取執行個體中繼資料](#)。

i Tip

若要在執行個體重新啟動時執行使用者資料指令碼，使用者資料必須採用 MIME 多部分格式，並在使用者資料的 `#cloud-config` 區段中指定以下內容：

```
#cloud-config
cloud_final_modules:
  - [scripts-user, always]
```

- 對於需要等服務的進階案例 AWS Lambda，當執行個體進入或離開暖集區時，您可以為 Auto Scaling 群組建立生命週期關聯，並設定目標服務以根據生命週期通知執行自訂動作。如需詳細資訊，請參閱[受支援的通知目標](#)。

執行個體休眠的準備作業

若要準備 Auto Scaling 執行個體以使用 Hibernated 集區狀態，請建立正確設定的新啟動範本或啟動組態以支援執行個體休眠，如 Amazon EC2 使用者指南中的[休眠先決條件](#)主題所述。然後，將新的啟動範本或啟動組態與 Auto Scaling 群組相關聯，並開始執行個體重新整理，以取代與之前的啟動範本或啟動組態關聯的執行個體。如需詳細資訊，請參閱[使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。

更新暖集區中的執行個體

若要更新暖集區中的執行個體，您可以建立新的啟動範本或啟動組態，並將其與 Auto Scaling 群組建立關聯。所有新的執行個體都會使用新的 AMI 和在啟動範本或啟動組態中指定的其他更新來啟動，但現有的執行個體不會受到影響。

若要強制使用新啟動範本或啟動組態的取代暖集區執行個體啟動，您可以啟動執行個體重新整理對群組進行滾動更新。執行個體重新整理會先取代 InService 執行個體。接著，其會取代暖集區中的執行個體。如需詳細資訊，請參閱[使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。

相關資源

您可以造訪我們的 [GitHub 儲存庫](#)，以取得暖集區的生命週期關聯範例。

限制

- 具有混合執行個體類型的 Auto Scaling 群組的暖集區限制：

- 加權混合執行個體群組不支援暖集區。如果您的 Auto Scaling 群組使用執行個體權重，則無法新增暖集區。
- 暖集區不支援混合執行個體群組中的 Spot 執行個體。只有在使用暖集區時，才能為隨需執行個體設定混合執行個體政策。
- 將暖集區與處於休眠狀態的混合執行個體群組搭配使用時，您必須在啟動範本 HibernationOptions 中設定。
- 只有當執行個體的根裝置為 Amazon EBS 磁碟區時，Amazon EC2 Auto Scaling 才可以讓執行個體處於 Stopped 或 Hibernated 狀態。無法將使用根裝置執行個體存放區的執行個體停止或休眠。
- 只有在 Amazon EC2 使用者指南中符合[休眠先決條件](#)主題中列出的所有要求時，Amazon EC2 Auto Scaling 才能將執行個體置於 Hibernated 狀態。Amazon EC2
- 如果暖集區在發生擴增事件時已耗盡，執行個體會直接啟動至 Auto Scaling 群組 (冷啟動)。如果可用區域的容量不足，您也可能遭遇冷啟動。
- 如果暖集區中的執行個體在啟動程序期間遇到問題，導致無法達到 InService 狀態，則該執行個體會被視為啟動失敗並終止。無論根本原因為何，例如容量不足錯誤或任何其他因素，這都適用。
- 如果嘗試透過 Amazon Elastic Kubernetes Service (Amazon EKS) 受管節點群組使用暖集區，則仍在初始化的執行個體可能會在 Amazon EKS 叢集中註冊。因此，當執行個體準備停止或休眠時，叢集可能會在執行個體上安排作業。
- 同樣，如果您嘗試在 Amazon ECS 叢集中使用暖集區，執行個體可能會在完成初始化之前向叢集註冊。若要解決此問題，您必須設定啟動範本或啟動組態，其中包含使用者資料中的特殊代理程式組態變數。如需詳細資訊，請參閱 Amazon Elastic Container Service 開發人員指南中的[使用 Auto Scaling 群組的暖集區](#)。

在 Auto Scaling 群組中使用具有暖集區的生命週期掛鉤

暖集區中的執行個體會維持其獨立生命週期，以便協助您為每個轉換建立適當的自訂動作。此生命週期旨在協助您在執行個體仍在初始化期間和投入使用之前，叫用目標服務 (例如 Lambda 函數) 中的動作。

Note

用於新增和管理 lifecycle hook 和完成生命週期動作的 API 操作不會變更。僅變更執行個體生命週期。

如需有關新增 lifecycle hook 的詳細資訊，請參閱 [將生命週期掛鉤新增至 Auto Scaling 群組](#)。如需有關完成生命週期動作的詳細資訊，請參閱 [在 Auto Scaling 群組中完成生命週期動作](#)。

對於進入暖集區的執行個體，由於以下原因之一，您可能需要 lifecycle hook：

- 您希望從需要很長時間才能完成初始化的 AMI 啟動 EC2 執行個體。
- 您希望執行使用者資料指令碼來引導 EC2 執行個體。

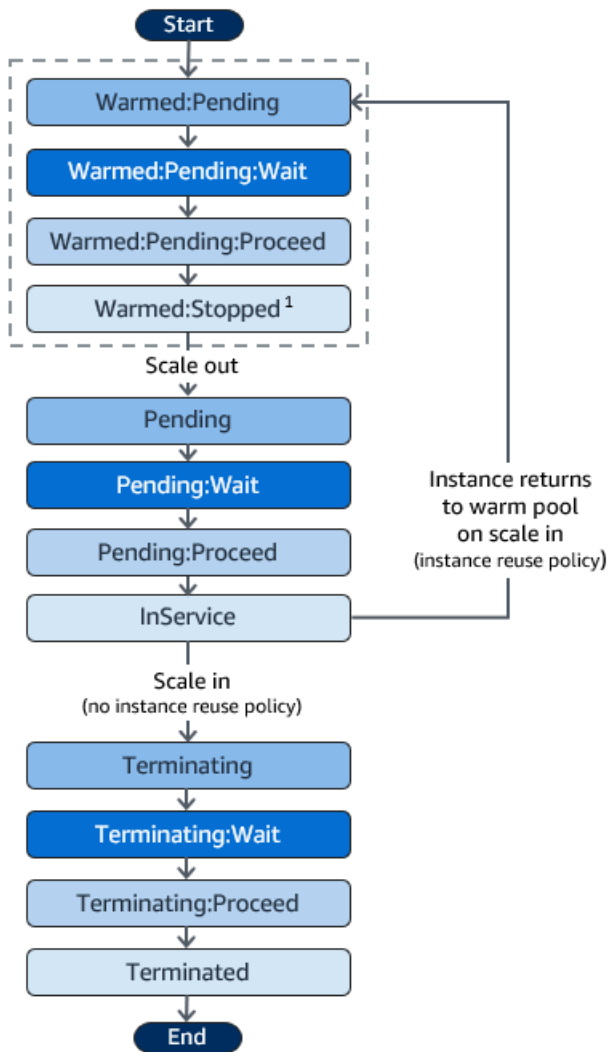
對於離開暖集區的執行個體，由於以下原因之一，您可能需要 lifecycle hook：

- 您可以使用一些額外的時間來準備 EC2 執行個體以供使用。例如，您可能需要服務必須在執行個體重啟時啟動，然後您的應用程式才能正常運作。
- 您要預先填入快取資料，使新伺服器不會以空白的快取進行啟動。
- 您要使用組態管理服務，將新執行個體註冊為受管理執行個體。

暖集區中的執行個體生命週期狀態轉換

作為其生命週期的一部分，Auto Scaling 執行個體可以在多種狀態之間轉換。

以下圖表顯示使用暖集區時 Auto Scaling 狀態之間的轉換：



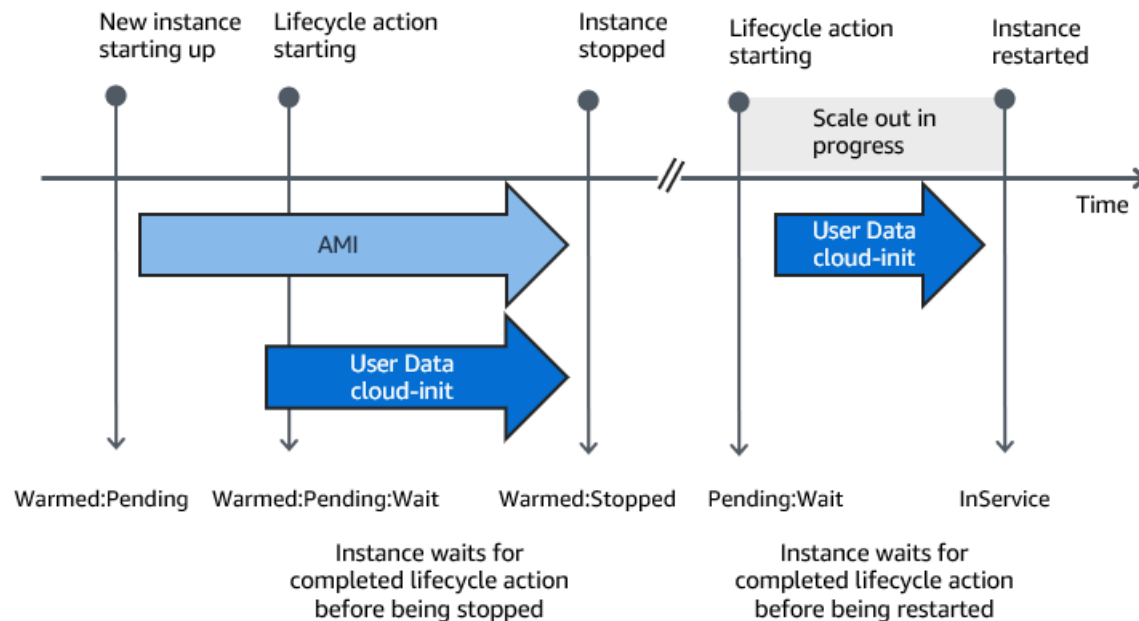
¹ 此狀態根據暖集區的集區狀態設定而異。如果集區狀態設定為 Running，則此狀態是 Warmed:Running。如果集區狀態設定為 Hibernated，則此狀態是 Warmed:Hibernated。

在新增 lifecycle hook 時，請考慮以下事項：

- 當為 autoscaling:EC2_INSTANCE_LAUNCHING 生命週期動作設定了 lifecycle hook 時，新啟動的執行個體會在達到 Warmed:Pending:Wait 狀態時先暫停以執行自訂動作，然後在執行個體重新啟動並達到 Pending:Wait 狀態時再次暫停。
- 當為 EC2_INSTANCE_TERMINATING 生命週期動作設定了 lifecycle hook 時，終止的執行個體在達到 Terminating:Wait 狀態時會暫停以執行自訂動作。不過，如果您指定執行個體重複使用政策，以便在縮減時將執行個體傳回暖集區，而不是終止執行個體，則傳回暖集區的執行個體會暫停，以便在 EC2_INSTANCE_TERMINATING 生命週期動作的 Warmed:Pending:Wait 狀態下執行自訂動作。

- 如果應用程式的需求將暖集區用盡，則只要群組尚未達到其最大容量，Amazon EC2 Auto Scaling 即可將執行個體直接啟動至 Auto Scaling 群組。如果執行個體直接啟動至群組，則這些執行個體只會在 Pending:Wait 狀態時暫停以執行自訂動作。
- 若要控制執行個體在轉換到下一個狀態之前保持等待狀態的時間長度，請將自訂動作設定為使用 complete-lifecycle-action 命令。有 lifecycle hook 時，執行個體會一直處於等待狀態，直到您通知 Amazon EC2 Auto Scaling 指定的生命週期動作已完成，或直到逾時期間結束 (預設為一小時)。

以下概述了橫向擴展事件的流程。



當執行個體進入等待狀態時，Amazon EC2 Auto Scaling 會傳送通知。有關這些通知的範例，請參閱本指南的「EventBridge」章節。如需詳細資訊，請參閱[暖集區範例事件和模式](#)。

受支援的通知目標

Amazon EC2 Auto Scaling 支援將以下任意內容定義為生命週期通知的通知目標：

- EventBridge 規則
- Amazon SNS 主題
- Amazon SQS 佇列
- AWS Lambda 函數

⚠ Important

如果您的啟動範本或啟動組態中有使用者資料 (cloud-init) 指令碼，可在執行個體啟動時設定執行個體，您不需要接收通知，即可在啟動或重新啟動的執行個體上執行自訂動作。

以下各節包含描述如何設定通知目標的文件連結：

EventBridge 規則 — 若要在 Amazon EC2 Auto Scaling 讓執行個體進入等待狀態時執行程式碼，您可以建立 EventBridge 規則，並將 Lambda 函數指定為其目標。若要根據不同的生命週期通知叫用不同的 Lambda 函數，您可以建立多個規則，並將每個規則與特定的事件模式和 Lambda 函數相關聯。如需詳細資訊，請參閱[為暖集區事件建立 EventBridge 規則](#)。

Amazon SNS 主題 — 若要在執行個體進入等待狀態時接收通知，您可以建立 Amazon SNS 主題，然後設定 Amazon SNS 訊息篩選，根據訊息屬性以不同的方式傳遞生命週期通知。如需詳細資訊，請參閱[使用 Amazon SNS 接收通知](#)。

Amazon SQS 佇列 — 若要設定生命週期通知的交付點，相關取用者可以在其中接收和處理它們，您可以建立 Amazon SQS 佇列和處理來自 SQS 佇列的訊息的佇列取用者。如果希望佇列取用者根據訊息屬性以不同方式處理生命週期通知，您也必須設定佇列取用者來解析訊息，然後在特定屬性符合所需值時對訊息採取行動。如需詳細資訊，請參閱[使用 Amazon SQS 接收通知](#)。

AWS Lambda 函數 — 若要在 Amazon EC2 Auto Scaling 讓執行個體進入等待狀態時執行自訂程式碼，您可以將 Lambda 函數指定為通知目標。Lambda 函數使用生命週期通知資料叫用，可讓您執行自訂動作，例如執行個體組態、應用程式設定，或與其他 AWS 服務的整合。您必須設定 Lambda 函數的資源型政策，以允許 Auto Scaling 服務連結角色叫用函數。如需詳細資訊，請參閱[將通知 AWS Lambda 直接路由到](#)。

為 Auto Scaling 群組建立暖集區

本主題說明如何為 Auto Scaling 群組建立暖集區。

⚠ Important

繼續之前，請完成建立暖集區的[先決條件](#)，並確認您已經為 Auto Scaling 群組建立 lifecycle hook。

建立暖集區

使用下列步驟來為 Auto Scaling 群組建立暖集區。

建立暖集區 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

頁面底部會開啟一個分割窗格。

3. 選擇 Instance management (執行個體管理) 索引標籤。
4. 在 Warm pool (暖集區) 中，選擇 Create warm pool (建立暖集區)。
5. 若要設定暖集區，請執行以下動作：
 - a. 對於 Warm pool instance state (暖集區執行個體狀態)，請選擇執行個體在進入暖集區時要轉換為什麼狀態。預設值為 Stopped。
 - b. 對於 Minimum warm pool size (暖集區大小下限)，輸入要在暖集區中維持的執行個體數目下限。
 - c. 對於執行個體重複使用，選取縮減時重複使用核取方塊，以允許 Auto Scaling 群組中的執行個體在縮減時返回暖集區。
 - d. 針對暖集區大小，選擇其中一個可用的選項：
 - 預設規格：暖集區的大小取決於 Auto Scaling 群組的最大容量和所需容量之間的差異。此選項可簡化暖集區管理。建立暖集區之後，只要調整群組的最大容量，即可輕鬆更新其大小。
 - 自訂規格：暖集區的大小取決於自訂值與 Auto Scaling 群組所需容量之間的差異。此選項可讓您獨立於群組的最大容量，靈活地管理暖集區的大小。
6. 檢視根據目前設定預估的暖集區大小區段，以確認預設或自訂規格如何套用至暖集區的大小。請記住，暖集區大小取決於 Auto Scaling 群組的所需容量，這會在群組擴展時變更。
7. 選擇建立。

混合執行個體群組的執行個體類型選取

當您的群組設定混合執行個體政策時，Auto Scaling 會在擴展事件期間優先考慮已在暖集區中的執行個體類型。啟動行為：

1. Auto Scaling 會嘗試使用暖集區中可用的執行個體類型啟動執行個體。
2. 如果暖啟動失敗，Auto Scaling 會使用混合執行個體政策中的所有剩餘執行個體類型來嘗試冷啟動。

Example

範例

如果您將 Auto Scaling 群組設定為 10 種執行個體類型，且暖集區包含其中 6 種執行個體類型。在向外擴展期間，Auto Scaling 會先嘗試暖集區的 6 種執行個體類型。如果失敗，Auto Scaling 會透過冷啟動嘗試所有設定的執行個體類型。

這可讓您盡可能暖集區效能優勢，同時維持完整混合執行個體組態的彈性。

刪除暖集區

如果不再需要暖集區，請按下列步驟將其刪除。

刪除暖集區 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

頁面底部會開啟一個分割窗格。

3. 選擇 Instance management (執行個體管理) 索引標籤。
4. 對於 Warm pool (暖集區)，選擇 Actions (動作)、Delete (刪除)。
5. 出現確認提示時，請選擇刪除。

檢視運作狀態檢查狀態和運作狀態檢查失敗的原因

運作狀態檢查允許 Amazon EC2 Auto Scaling 判斷執行個體是否運作狀態不佳且應遭到終止。若為保持在 Stopped 狀態的暖集區執行個體，其會採用 Amazon EBS 具有的 Stopped 執行個體可用性知識來識別運作狀態不佳的執行個體。其會藉由呼叫 DescribeVolumeStatus API 來判斷連接至執行個體的 EBS 磁碟區的狀態。若為保持在 Running 狀態的暖集區執行個體，其會依賴 EC2 狀態檢查來判斷執行個體的運作狀態。雖然暖集區執行個體沒有運作狀態檢查寬限期，但 Amazon EC2 Auto Scaling 在 lifecycle hook 完成之前，不會開始檢查執行個體運作狀態。

在執行個體運作狀態不佳時，Amazon EC2 Auto Scaling 會自動刪除運作狀態不佳的執行個體，並建立一個新的執行個體來取代它。執行個體在運作狀態檢查失敗後，通常會在幾分鐘內遭到終止。如需詳細資訊，請參閱[檢視運作狀態檢查失敗的原因](#)。

亦支援自訂運作狀態檢查。如果您有自有運作狀態檢查系統可以偵測執行個體的運作狀態，並將此資訊傳送至 Amazon EC2 Auto Scaling，這頗有助益。如需詳細資訊，請參閱[為您的 Auto Scaling 群組設定自訂運作狀態檢查](#)。

在 Amazon EC2 Auto Scaling 主控台中，您可以檢視暖集區執行個體的状态 (運作狀態良好或運作狀態不佳)。您也可以使用 AWS CLI 或其中一個 SDKs 檢視其運作狀態。

檢視暖集區執行個體的状态 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Instance management (執行個體管理) 索引標籤的 Warm pool Instances (暖集區執行個體) 中，Lifecycle (生命週期) 資料欄會包含執行個體的状态。

Health status (運作狀態) 資料欄中會顯示 Amazon EC2 Auto Scaling 對執行個體運作狀態進行的評估。

Note

新執行個體以良好的運作狀態開始啟動。在 lifecycle hook 完成之前，系統不會檢查執行個體的運作狀態。

檢視運作狀態檢查失敗的原因 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 中，Status (狀態) 欄位會顯示 Auto Scaling 群組是否已成功啟動或終止執行個體。

若其已終止任何運作狀態不佳的執行個體，Cause (原因) 資料欄會顯示終止的日期和時間，以及運作狀態檢查失敗的原因。例如，At 2021-04-01T21:48:35Z an instance was taken out of service in response to EBS volume health check failure (在 2021-04-01T 21:48:35 Z 時，執行個體因 EBS 磁碟區運作狀態檢查失敗而停止運作)。

檢視暖集區執行個體的状态 (AWS CLI)

使用以下 [describe-warm-pool](#) 命令來檢視 Auto Scaling 群組的暖集區。

```
aws autoscaling describe-warm-pool --auto-scaling-group-name my-asg
```

範例輸出。

```
{
  "WarmPoolConfiguration": {
    "MinSize": 0,
    "PoolState": "Stopped"
  },
  "Instances": [
    {
      "InstanceId": "i-0b5e5e7521cf4a46c",
      "InstanceType": "t2.micro",
      "AvailabilityZone": "us-west-2a",
      "LifecycleState": "Warmed:Stopped",
      "HealthStatus": "Healthy",
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "Version": "1"
      }
    },
    {
      "InstanceId": "i-0e21af9dcfb7aa6bf",
      "InstanceType": "t2.micro",
      "AvailabilityZone": "us-west-2a",
      "LifecycleState": "Warmed:Stopped",
      "HealthStatus": "Healthy",
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "Version": "1"
      }
    }
  ]
}
```



```

    }
  }
]
}

```

檢視運作狀態檢查失敗的原因 (AWS CLI)

使用以下 [describe-scaling-activities](#) 命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下為回應範例，其中 Description 表示您的 Auto Scaling 群組已終止執行個體，而 Cause 則指出運作狀態檢查失敗的原因。

擴展活動依開始時間排序。首先描述仍在進行中的活動。

```

{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in response to EBS volume health check failure.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:283179a2-f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
    ...
  ]
}

```

使用 建立和管理暖集區的範例 AWS CLI

您可以使用 AWS 管理主控台、AWS Command Line Interface (AWS CLI) 或 SDKs 建立和管理暖集區。

以下範例說明如何使用 AWS CLI 建立和管理暖集區。

目錄

- [範例 1：將執行個體保持為 Stopped 狀態](#)
- [範例 2：將執行個體保持為 Running 狀態](#)
- [範例 3：將執行個體保持為 Hibernated 狀態](#)
- [範例 4：縮減時將執行個體傳回暖集區](#)
- [範例 5：指定暖集區中的執行個體數量下限](#)
- [範例 6：使用自訂規格定義暖集區大小](#)
- [範例 7：定義暖集區絕對大小](#)
- [範例 8：刪除暖集區](#)

範例 1：將執行個體保持為 **Stopped** 狀態

以下 [put-warm-pool](#) 範例會建立一個暖集區，可將執行個體維持在 Stopped 狀態。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped
```

範例 2：將執行個體保持為 **Running** 狀態

以下 [put-warm-pool](#) 範例會建立一個暖集區，可將執行個體維持在 Running 狀態，而不是 Stopped 狀態。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Running
```

範例 3：將執行個體保持為 **Hibernated** 狀態

以下 [put-warm-pool](#) 範例會建立一個暖集區，可將執行個體維持在 Hibernated 狀態，而不是 Stopped 狀態。這讓您可以停止執行個體而不刪除其記憶體內容 (RAM)。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Hibernated
```

範例 4：縮減時將執行個體傳回暖集區

以下 [put-warm-pool](#) 範例會建立一個暖集區，可將執行個體維持在 Stopped 狀態，並包含 `--instance-reuse-policy` 選項。執行個體重複使用政策值 `'{"ReuseOnScaleIn": true}'` 會告訴 Amazon EC2 Auto Scaling 在 Auto Scaling 群組縮減時將執行個體傳回暖集區。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --instance-reuse-policy '{"ReuseOnScaleIn": true}'
```

範例 5：指定暖集區中的執行個體數量下限

以下 [put-warm-pool](#) 範例建立一個至少維護 4 個執行個體的暖集區，使至少有 4 個執行個體可用於處理流量峰值。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --min-size 4
```

範例 6：使用自訂規格定義暖集區大小

根據預設，Amazon EC2 Auto Scaling 會管理暖集區的大小，作為 Auto Scaling 群組的最大容量和所需容量之間的差異。不過，您可以使用 `--max-group-prepared-capacity` 選項，獨立於群組的最大容量來管理暖集區的大小。

下列 [put-warm-pool](#) 範例會建立暖集區，並設定暖集區和 Auto Scaling 群組中可同時存在的執行個體數量上限。如果群組的所需容量為 800，則暖集區在執行此命令後初始化時，最初的大小為 100。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --max-group-prepared-capacity 900
```

若要維持暖集區中的執行個體數目下限，請使用命令納入 `--min-size` 選項，如下所示。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --max-group-prepared-capacity 900 --min-size 25
```

範例 7：定義暖集區絕對大小

如果將 `--max-group-prepared-capacity` 和 `--min-size` 選項設定為相同的值，則暖集區具有絕對大小。以下 [put-warm-pool](#) 範例會建立一個維持恆定暖集區大小 (包含 10 個執行個體) 的暖集區。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --min-size 10 --max-group-prepared-capacity 10
```

範例 8：刪除暖集區

使用下列 [delete-warm-pool](#) 命令來刪除暖集區。

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg
```

如果暖集區中具有執行個體，或有擴展活動正在進行中，請合併使用 [delete-warm-pool](#) 命令與 `--force-delete` 選項。此選項也會終止 Amazon EC2 執行個體和任何未完成的生命週期動作。

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg --force-delete
```

Auto Scaling 群組區域轉移

區域轉移是 Amazon Application Recovery Controller (ARC) 中的功能。使用區域轉移，您可以透過單一動作，快速從可用區域中的應用程式受損復原。當您為 Auto Scaling 群組啟用區域轉移時，該群組會向 ARC 區域轉移服務註冊。然後，您可以使用 AWS 管理主控台或 API 啟動區域轉移 AWS CLI，而 Auto Scaling 群組會將具有作用中區域轉移的可用區域視為受損。

Auto Scaling 群組區域轉移概念

在繼續之前，請確定您已熟悉與 ARC 區域轉移整合相關的下列核心概念。

ARC 區域轉移

當您啟用此功能時，Auto Scaling 可以向 ARC 區域轉移註冊 Auto Scaling 群組。註冊後，您可以使用 [ARC ListManagedResources](#) API 檢視資源。如需詳細資訊，請參閱《Amazon Application Recovery Controller (ARC) 開發人員指南》中的 [ARC 中的區域轉移](#)。

可用區域重新平衡

Auto Scaling 會嘗試在每個可用區域中保持容量平衡。當可用區域之間發生不平衡時，Auto Scaling 會自動嘗試修正不平衡。如需詳細資訊，請參閱[執行個體分佈](#)。

動態擴展

動態擴展會根據您使用擴展政策選擇的指標，擴展 Auto Scaling 群組所需的容量。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 動態擴展](#)。

運作狀態檢查

Auto Scaling 會定期檢查 Auto Scaling 群組內所有執行個體的運作狀態，以確保它們正在執行且狀態良好。偵測到運作狀態不佳的執行個體時，Auto Scaling 會將其標記為取代。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

執行個體重新整理

您可以使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體。啟動執行個體重新整理後，Auto Scaling 會嘗試取代 Auto Scaling 群組中的所有執行個體。如需詳細資訊，請參閱[使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。

預先擴展

您可以容忍單一可用區域的遺失，因為您在應用程式的剩餘可用區域中有足夠的容量。

向外擴展

當您增加 Auto Scaling 群組的所需容量時，Auto Scaling 會嘗試啟動其他執行個體以符合新的所需容量。根據預設，Auto Scaling 會以平衡的方式啟動執行個體，以維持 Auto Scaling 群組中每個已啟用可用區域的相同容量。

Auto Scaling 群組的區域轉移運作方式

假設您有一個具有下列可用區域的 Auto Scaling 群組：

- us-east-1a
- us-east-1b
- us-east-1c

您已在所有可用區域中啟用區域轉移，並在 中注意到失敗，us-east-1a因此您觸發區域轉移。在 中觸發區域轉移時，會發生下列行為us-east-1a。

- 向外擴展 – Auto Scaling 將在運作狀態良好的可用區域 (us-east-1b 和) 中啟動所有新的容量請求us-east-1c。
- 動態擴展 – Auto Scaling 會封鎖擴展政策，防止所有可用區域中所需的容量減少。Auto Scaling 不會封鎖擴展政策在所有可用區域中增加所需容量。
- 執行個體重新整理 – Auto Scaling 將延長區域轉移作用中時延遲的任何執行個體重新整理程序的逾時。

下表說明在 中觸發區域轉移時，每個選項的運作狀態檢查行為us-east-1a。

可用區域運作狀態檢查行為選擇受損	運作狀態檢查行為			
取代運作狀態不佳	在所有可用區域 (us-east-1a 、 us-east-1b 和) 中，會取代運作狀態不佳的執行個體us-east-1c 。			
忽略運作狀態不佳	在 us-east-1b 和 中，會取代看起來運作狀態不佳的執行個體us-east-1c 。執行個體不會在可用區域中取代為作用中區域轉移 (us-east-1a)。			

使用區域轉移的最佳實務

若要在使用區域轉移時維持應用程式的高可用性，我們建議下列最佳實務：

- 監控 EventBridge 通知，以判斷何時發生持續的可用區域受損事件。如需詳細資訊，請參閱[使用 EventBridge 處理 Auto Scaling 事件](#)。
- 使用具有適當閾值的擴展政策，以確保您有足夠的容量來容忍遺失可用區域。
- 設定運作狀態最低百分比為 100 的執行個體維護政策。使用此設定，Auto Scaling 會等待新的執行個體準備就緒，再終止運作狀態不佳的執行個體。

對於預先擴展的客戶，我們也建議執行下列動作：

- 選取忽略運作狀態不佳做為受損可用區域的運作狀態檢查行為，因為您在受損事件期間不需要取代運作狀態不佳的執行個體。
- 在 ARC 中為您的 Auto Scaling 群組使用區域自動轉移。ARC 中的區域自動轉移功能允許在 AWS 偵測到可用區域中的損害時，將資源的流量 AWS 移離可用區域。如需詳細資訊，請參閱《Amazon Application Recovery Controller (ARC) 開發人員指南》中的 [ARC 中的區域自動轉移](#)。

對於具有跨區域停用負載平衡器的客戶，我們也建議執行下列動作：

- 僅對您的可用區域分佈使用平衡。
- 如果您在 Auto Scaling 群組和負載平衡器上使用區域轉移，請先取消 Auto Scaling 群組上的區域轉移。然後，在取消負載平衡器上的區域轉移之前，等待容量在所有可用區域之間平衡。
- 由於啟用區域轉移並使用跨區域停用的負載平衡器時，容量可能會不平衡，因此 Auto Scaling 包含額外的驗證步驟。如果您遵循最佳實務，您可以透過選取 AWS 管理主控台 核取方塊或使用 `CreateAutoScalingGroup`、或 `UpdateAutoScalingGroup` 中的 `skip-zonal-shift-validation` 旗標來確認此可能性 `AttachTrafficSources`。

如需搭配 Auto Scaling 群組使用區域轉移的詳細資訊，請參閱 [搭配 Amazon EC2 Auto Scaling 使用區域轉移](#) AWS 的運算部落格。

使用 AWS 管理主控台 或 啟用區域轉移 AWS CLI

若要啟用區域轉移，請使用下列其中一種方法。

Console

在新群組上啟用區域轉移（主控台）

1. 遵循 中的指示 [使用啟動範本建立 Auto Scaling 群組](#) 並完成程序中的每個步驟，直到步驟 10。
2. 在與其他服務整合頁面上，針對應用程式復原控制器 (ARC) 區域轉移，選取核取方塊以啟用區域轉移。
3. 針對運作狀態檢查行為，選擇忽略運作狀態不佳或取代運作狀態不佳。如需詳細資訊，請參閱 [Auto Scaling 群組的區域轉移運作方式](#)。
4. 繼續 [使用啟動範本建立 Auto Scaling 群組](#) 中的步驟。

AWS CLI

在新群組上啟用區域轉移 (AWS CLI)

將 `--availability-zone-impairment-policy` 參數新增至 [create-auto-scaling-group](#) 命令。

`--availability-zone-impairment-policy` 參數有兩個選項：

- `ZonalShiftEnabled` – 如果設定為 `true`，Auto Scaling 會使用 ARC 區域轉移註冊 Auto Scaling 群組，您可以在 ARC 主控台上[啟動、更新或取消區域轉移](#)。如果設定為 `false`，Auto Scaling 會從 ARC 區域轉移取消註冊 Auto Scaling 群組。您必須已啟用區域轉移，才能將設定為 `false`。
- `ImpairedZoneHealthCheckBehavior` – 如果設定為 `replace-unhealthy`，則運作狀態不佳的執行個體將在可用區域中以作用中區域轉移取代。如果設為 `ignore-unhealthy`，則運作狀態不佳的執行個體不會在可用區域中以作用中區域轉移取代。如需詳細資訊，請參閱[Auto Scaling 群組的區域轉移運作方式](#)。

下列範例會在名為 `my-asg` 的新 Auto Scaling 群組上啟用區域轉移。

```
aws autoscaling create-auto-scaling-group \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --auto-scaling-group-name my-asg \
  --min-size 1 \
  --max-size 10 \
  --desired-capacity 5 \
  --availability-zones us-east-1a us-east-1b us-east-1c \
  --availability-zone-impairment-policy '{
    "ZonalShiftEnabled": true,
    "ImpairedZoneHealthCheckBehavior": IgnoreUnhealthy
  }'
```

Console

在現有群組上啟用區域轉移（主控台）

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。

2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所在的 AWS 區域。
3. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在整合索引標籤的應用程式復原控制器 (ARC) 區域轉移下，選擇編輯。
5. 選取核取方塊以啟用區域轉移。
6. 針對運作狀態檢查行為，選擇忽略運作狀態不佳或取代運作狀態不佳。如需詳細資訊，請參閱[Auto Scaling 群組的區域轉移運作方式](#)。
7. 選擇更新。

AWS CLI

在現有群組上啟用區域轉移 (AWS CLI)

將 `--availability-zone-impairment-policy` 參數新增至 [update-auto-scaling-group](#) 命令。

`--availability-zone-impairment-policy` 參數有兩個選項：

- `ZonalShiftEnabled` – 如果設定為 `true`，Auto Scaling 會使用 ARC 區域轉移註冊 Auto Scaling 群組，您可以在 ARC 主控台上[啟動、更新或取消區域轉移](#)。如果設定為 `false`，Auto Scaling 會從 ARC 區域轉移取消註冊 Auto Scaling 群組。您必須已啟用區域轉移，才能將設定為 `false`。
- `ImpairedZoneHealthCheckBehavior` – 如果設定為 `replace-unhealthy`，運作狀態不佳的執行個體將在可用區域中以作用中區域轉移取代。如果設為 `ignore-unhealthy`，則運作狀態不佳的執行個體不會在可用區域中以作用中區域轉移取代。如需詳細資訊，請參閱[Auto Scaling 群組的區域轉移運作方式](#)。

下列範例會在指定的 Auto Scaling 群組上啟用區域轉移。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --availability-zone-impairment-policy '{
    "ZonalShiftEnabled": true,
    "ImpairedZoneHealthCheckBehavior": IgnoreUnhealthy
  }'
```

Auto Scaling 群組可用區域分佈

以下資訊說明 Auto Scaling 群組可用區域策略。

平衡的努力

Auto Scaling 會在已啟用的可用區域中維持相同數量的執行個體。如果啟動嘗試在可用區域中失敗，Auto Scaling 會嘗試在另一個運作狀態良好的可用區域中啟動執行個體。此策略對於需要可用區域備援的應用程式很重要，而且不會受到不平衡群組的影響。

僅平衡

Auto Scaling 會在已啟用的可用區域中維持相同數量的執行個體。如果啟動嘗試在可用區域中失敗，Auto Scaling 將繼續嘗試在可用區域中啟動執行個體。此策略對於滿足規定人數型工作負載等特定需求，或您的 Auto Scaling 群組可以容忍可用區域遺失，因為您在剩餘可用區域中有足夠的容量。

可用區域分佈策略選擇位於 的網路區段中，AWS 管理主控台 或者您可以使用 [create-auto-scaling-group](#) 或 [update-auto-scaling-group](#) 命令。

如需詳細資訊，請參閱[使用啟動範本建立 Auto Scaling 群組](#)。

從 Auto Scaling 群組分離或連接執行個體

您可以從 Auto Scaling 群組分離執行個體。分離執行個體後，該執行個體會變得獨立，並且可以自行管理或連接到不同的 Auto Scaling 群組，與所屬的原始群組分開。例如，當您想要使用已經在執行應用程式的現有執行個體執行測試時，這會很有用。

本主題提供如何分離和連接執行個體的說明。連接執行個體時，您也可以使用現有的執行個體，而不是分離的執行個體。

我們建議您使用待命程序暫時將執行個體從群組中移除，而不是將執行個體分離並重新連接至相同的群組。如需詳細資訊，請參閱[從 Auto Scaling 群組暫時移除執行個體](#)。

目錄

- [分離執行個體的考量事項](#)
- [連接執行個體的考量事項](#)
- [使用分離和連接將執行個體移至不同的群組](#)

分離執行個體的考量事項

當您分離執行個體時，請記住以下幾點：

- 只有當執行個體處於 InService 或 StandBy 狀態時，才能將其分離。如果您要分離位於 StandBy 狀態的執行個體，請務必謹慎。在將執行個體置於 StandBy 狀態之後嘗試分離執行個體時，在 API 呼叫中包含 ShouldDecrementDesiredCapacity 旗標，可能會導致其他執行個體意外終止。
- 分離執行個體之後，執行個體會繼續執行並產生費用。為了避免不必要的費用，請確保在不再需要分離的執行個體時重新連接或終止分離的執行個體。
- 您可以選擇依您要分離的執行個體數量減少所需的容量。如果您選擇不減少容量，Amazon EC2 Auto Scaling 會啟動新的執行個體來取代分離的執行個體，以維持所需的容量。
- 如果您要分離的執行個體數量會使 Auto Scaling 群組低於其容量下限，則必須減少容量下限。
- 如果您從相同的可用區域分離多個執行個體，而不減少所需的容量，則除非您暫停 AZRebalance 程序，否則群組會自行重新平衡。如需詳細資訊，請參閱 [暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。
- 如果是從已連接負載平衡器目標群組或 Classic Load Balancer 的 Auto Scaling 群組分開執行個體，則執行個體會從負載平衡器取消註冊。如果為您的負載平衡器啟用連接耗盡 (取消註冊延遲)，Amazon EC2 Auto Scaling 會等待處理中的請求完成。

連接執行個體的考量事項

連接執行個體時，請注意下列事項：

- Amazon EC2 Auto Scaling 會將連接的執行個體視為與群組本身啟動的執行個體相同。這表示如果已選取連接執行個體，則可以在縮減事件期間將其終止。AWSServiceRoleForAutoScaling 服務連結角色授予的許可允許 Amazon EC2 Auto Scaling 這樣做。
- 當您連接執行個體，所需的容量會以連接中的執行個體的數量來增加。如果在新增執行個體之後所需的容量超過群組的大小上限，連接更多執行個體的請求會失敗。
- 如果您將執行個體新增至群組，導致跨可用區域分佈不均，Amazon EC2 Auto Scaling 會重新平衡群組，以重新建立均勻分佈，除非您暫停 AZRebalance 程序。如需詳細資訊，請參閱 [暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。
- 如果您將執行個體連接到已連接負載平衡器目標群組或 Classic Load Balancer 的 Auto Scaling 群組，則執行個體會註冊到負載平衡器。

要連接的執行個體必須滿足下列條件：

- 執行個體處於 Amazon EC2 的 `running` 狀態。
- 使用 AMI 啟動的執行個體仍然必須存在。
- 該執行個體不屬於另一個 Auto Scaling 群組。
- 執行個體會在 Auto Scaling 群組中定義的其中一個可用區域中啟動。
- 如果 Auto Scaling 群組已連接了負載平衡器目標群組或 Classic Load Balancer，則執行個體和負載平衡器必須同時都在相同的 VPC 中。

使用分離和連接將執行個體移至不同的群組

使用下列其中一個程序，從 Auto Scaling 群組分離執行個體，並將其連接至不同的 Auto Scaling 群組。

若要從分離的執行個體建立新的 Auto Scaling 群組，請參閱 [使用從現有執行個體建立 Auto Scaling 群組 AWS CLI](#)（不建議，會建立啟動組態）。

Console

將執行個體與 Auto Scaling 群組分開

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Instance management (執行個體管理) 索引標籤的 Instances (執行個體) 中，選取執行個體，再選擇 Actions (動作)、Detach (分開)。
4. 在分離執行個體對話方塊中，保持選取替代執行個體核取方塊，以啟動替代執行個體。清除此核取方塊以減少所需容量。
5. 出現確認提示時，請輸入 **detach** 來確認移除特定的 Auto Scaling 群組，然後選擇分離執行個體。

您現在可以將執行個體連接至不同的 Auto Scaling 群組。

將執行個體連接至 Auto Scaling 群組

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。

2. (選用) 在導覽窗格的 AUTO SCALING 中，選擇 Auto Scaling Groups (Auto Scaling 群組)。選擇 Auto Scaling 群組，然後確認 Auto Scaling 群組的大小上限是否足夠讓您新增其他執行個體。否則，在 Details (詳細資訊) 索引標籤上增加容量上限。
3. 在導覽窗格的 Instances (執行個體) 下選擇 Instances (執行個體)，然後選取執行個體。
4. 選擇 Actions (動作)、Instance settings (執行個體設定)、Attach to Auto Scaling Group (連接到 Auto Scaling 群組)。
5. 在 Attach to Auto Scaling Group (連接到 Auto Scaling 群組) 頁面上，針對 Auto Scaling Group (Auto Scaling 群組) 選擇 Auto Scaling 群組，然後選擇 Attach (連接)。
6. 如果執行個體不符合條件，您會得到一則包含詳細資訊的錯誤訊息。例如，執行個體可能並非與 Auto Scaling 群組位於相同的可用區域。選擇關閉，並使用符合條件的 Auto Scaling 群組再試一次。

AWS CLI

若要分離和連接執行個體，請使用下列範例命令。將每個#####替換為自己的資訊。

將執行個體與 Auto Scaling 群組分開

1. 若要描述目前的執行個體，請使用下列 [describe-auto-scaling-instances](#) 命令。

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

下列範例顯示執行此命令時的輸出。

請記下您要從群組中移除的執行個體 ID。在下一個步驟中，您需要此 ID。

```
{
  "AutoScalingInstances": [
    {
      "ProtectedFromScaleIn": false,
      "AvailabilityZone": "us-west-2a",
      "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
      },
      "InstanceId": "i-05b4f7d5be44822a6",
      "InstanceType": "t3.micro",
```

```

    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService"
  },
  {
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1",
      "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-0c20ac468fa3049e8",
    "InstanceType": "t3.micro",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService"
  },
  {
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1",
      "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-0787762faf1c28619",
    "InstanceType": "t3.micro",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService"
  },
  {
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1",
      "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-0f280a4c58d319a8a",
    "InstanceType": "t3.micro",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",

```

```

        "LifecycleState": "InService"
      }
    ]
  }
}

```

- 若要分離執行個體而不減少所需的容量，請使用下列 [detach-instances](#) 命令。

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg
```

若要分離執行個體並減少所需的容量，請包含 `--should-decrement-desired-capacity` 選項。

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

您現在可以將執行個體連接至不同的 Auto Scaling 群組。

將執行個體連接至 Auto Scaling 群組

- 若要將執行個體連接至不同的 Auto Scaling 群組，請使用下列 [attach-instances](#) 命令。

```
aws autoscaling attach-instances --instance-ids i-05b4f7d5be44822a6 --auto-
scaling-group-name my-asg-for-testing
```

- 若要在連接執行個體後驗證 Auto Scaling 群組的大小，請使用下列 [describe-auto-scaling-groups](#) 命令。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg-
for-testing
```

下列範例回應顯示 群組有兩個執行中的執行個體，其中一個是您連接的執行個體。

```

{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg-for-testing",
      "AutoScalingGroupARN": "arn",
      "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",

```

```

        "Version": "2",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 2,
    ...
    "Instances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "HealthStatus": "Healthy",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "2",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-00dcdfffd5175890",
            "InstanceType": "t3.micro",
            "HealthStatus": "Healthy",
            "LifecycleState": "InService"
        }
    ],
    ...
}
]
}

```


從 Auto Scaling 群組暫時移除執行個體

您可以讓處於 InService 狀態的執行個體進入 Standby 狀態，更新執行個體或疑難排解執行個體問題，然後讓執行個體恢復到服務狀態。處於待命狀態的執行個體仍然屬於 Auto Scaling 群組，但不會主動處理負載平衡器流量。

此功能可協助您停止並啟動執行個體或重新啟動執行個體，而不必擔心 Amazon EC2 Auto Scaling 會在進行運作狀態檢查時或縮減事件期間終止執行個體。

例如，您隨時可以透過變更啟動範本或啟動組態來變更 Auto Scaling 群組的 Amazon Machine Image (AMI)。Auto Scaling 群組啟動的任何後續執行個體都會使用此 AMI。不過，Auto Scaling 群組不會更新目前正在服務的執行個體。您可以終止這些執行個體並讓 Amazon EC2 Auto Scaling 取代它們，或使用執行個體重新整理功能來終止和取代執行個體。您也可以讓執行個體進入待命狀態、更新軟體，然後讓執行個體恢復到服務狀態。

從 Auto Scaling 群組中分開執行個體的程序與將執行個體置於待命狀態類似。如果您想要將執行個體連接到不同的群組，或像獨立 EC2 執行個體一樣管理執行個體並可能終止執行個體，分離執行個體可能很有用。如需詳細資訊，請參閱[從 Auto Scaling 群組分離或連接執行個體](#)。

目錄

- [待命狀態的運作方式](#)
- [考量事項](#)
- [待命狀態中的執行個體運作狀態](#)
- [將執行個體設定為待命，以暫時移除執行個體](#)

待命狀態的運作方式

待命狀態的運作方式如下所示，可協助您暫時從 Auto Scaling 群組移除執行個體：

1. 請您將該執行個體設為待命狀態。執行個體會維持此狀態，直到執行個體退出待命狀態。
2. 如果有負載平衡器目標群組或 Classic Load Balancer 連接到 Auto Scaling 群組，執行個體會從負載平衡器取消註冊。如果為負載平衡器啟用連接耗盡，Elastic Load Balancing 會依預設等待 300 秒，再完成取消註冊程序，這有助完成執行中的請求。
3. 您可以更新執行個體或對其進行故障排除。
4. 透過退出待命狀態可使執行個體恢復到服務狀態。
5. 如果有負載平衡器目標群組或 Classic Load Balancer 連接到 Auto Scaling 群組，執行個體會向負載平衡器註冊。

如需 Auto Scaling 群組中執行個體生命週期的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 執行個體生命週期](#)。

考量事項

下列是將執行個體移入待命狀態時的考量事項：

- 當您將執行個體置於待命狀態時，您可以透過此作業減少所需容量，也可以使其保持相同的值。
 - 如果您選擇不減少 Auto Scaling 群組的所需容量，則 Amazon EC2 Auto Scaling 會啟動執行個體以取代處於待命狀態的執行個體。其目的是當一個或多個執行個體處於待命狀態時，協助您維持應用程式的容量。
 - 如果您選擇減少 Auto Scaling 群組的所需容量，這樣可以防止啟動執行個體以取代處於待命狀態的執行個體。
- 執行個體恢復到服務狀態後，所需容量會增加，以反映 Auto Scaling 群組中的執行個體數。
- 若要執行增加 (和減少)，新的所需容量必須介於最小和最大群組大小之間。否則，操作會失敗。
- 如果在將執行個體置於待命狀態或透過退出待命狀態將執行個體返回服務狀態之後，發現 Auto Scaling 群組在可用區域之間無法平衡，Amazon EC2 Auto Scaling 會透過重新平衡可用區域來進行補償，除非您暫停 AZRebalance 程序。如需詳細資訊，請參閱 [暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。
- 您需支付待命狀態的執行個體費用。

待命狀態中的執行個體運作狀態

Amazon EC2 Auto Scaling 不會對待命狀態的執行個體執行運作狀態檢查。當執行個體處於待命狀態時，其顯示的運作狀態為進入待命狀態前的狀態。直到待命的執行個體恢復到服務狀態，Amazon EC2 Auto Scaling 才會對它們執行運作狀態檢查。

例如，如果讓狀態良好的執行個體進入待命狀態，再將其終止，Amazon EC2 Auto Scaling 仍會回報此執行個體狀態良好。如果嘗試讓處於待命狀態的已終止執行個體重新進入服務，Amazon EC2 Auto Scaling 會執行個體運作狀態檢查，若判斷為處於終止狀態且運作狀態不佳，則會啟動替換執行個體。如需詳細資訊，請參閱 [Auto Scaling 群組中的執行個體運作狀態檢查](#)。

將執行個體設定為待命，以暫時移除執行個體

使用下列其中一個程序，讓執行個體進入待命狀態，暫時停止服務。

Console

暫時移除執行個體

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Instance management (執行個體管理) 索引標籤的 Instances (執行個體) 中，選取執行個體。
4. 選擇 Actions (動作)、Set to Standby (設定為待命)。
5. 在設定為待命對話方塊中，保持選取替代執行個體核取方塊，以啟動替代執行個體。清除此核取方塊以減少所需容量。
6. 出現確認提示時，輸入 **standby** 以確認將指定的執行個體置於 Standby 狀態，然後選擇設定為待命。
7. 您可以視需求更新執行個體或排解問題。完成後，接續下一個步驟，讓執行個體恢復到服務狀態。
8. 選取執行個體，選擇 Actions (動作)、Set to InService (設為 InService)。在 Set to InService (設為 InService) 對話方塊中，選擇 Set to InService (設為 InService)。

AWS CLI

若要暫時從 Auto Scaling 群組中移除執行個體，請使用下列範例命令。將每個 **#####** 替換為自己的資訊。

暫時移除執行個體

1. 使用以下 [describe-auto-scaling-instances](#) 命令，找出需要更新的執行個體。

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

下列範例顯示執行此命令時的輸出。

請記下您要從群組中移除的執行個體 ID。在下一個步驟中，您需要此 ID。

```
{
```

```

    "AutoScalingInstances": [
      {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1",
          "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-05b4f7d5be44822a6",
        "InstanceType": "t3.micro",
        "AutoScalingGroupName": "my-asg",
        "HealthStatus": "HEALTHY",
        "LifecycleState": "InService"
      },
      ...
    ]
  }

```

2. 使用以下 [enter-standby](#) 命令，將執行個體設為 Standby 狀態。--should-decrement-desired-capacity 選項會減少所需容量，讓 Auto Scaling 群組不會啟動替代執行個體。

```

aws autoscaling enter-standby --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity

```

以下是回應範例。

```

{
  "Activities": [
    {
      "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",
      "AutoScalingGroupName": "my-asg",
      "Description": "Moving EC2 instance to Standby:
i-05b4f7d5be44822a6",
      "Cause": "At 2023-12-15T21:31:26Z instance i-05b4f7d5be44822a6 was
moved to standby
in response to a user request, shrinking the capacity from 4 to
3.",
      "StartTime": "2023-12-15T21:31:26.150Z",
      "StatusCode": "InProgress",
      "Progress": 50,
      "Details": "{\"Subnet ID\":\"subnet-c934b782\",\"Availability Zone
\": \"us-west-2a\"}"
    }
  ]
}

```

```
    }
  ]
}
```

3. (選用) 使用以下內容 [describe-auto-scaling-instances](#) 命令，確認執行個體位於 Standby。

```
aws autoscaling describe-auto-scaling-instances --instance-ids i-05b4f7d5be44822a6
```

以下是回應範例。請注意，現在執行個體的狀態為 Standby。

```
{
  "AutoScalingInstances": [
    {
      "ProtectedFromScaleIn": false,
      "AvailabilityZone": "us-west-2a",
      "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
      },
      "InstanceId": "i-05b4f7d5be44822a6",
      "InstanceType": "t3.micro",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "Standby"
    },
    ...
  ]
}
```

4. 您可以視需求更新執行個體或排解問題。完成後，接續下一個步驟，讓執行個體恢復到服務狀態。
5. 使用以下 [exit-standby](#) 命令，讓執行個體回到服務狀態。

```
aws autoscaling exit-standby --instance-ids i-05b4f7d5be44822a6 --auto-scaling-group-name my-asg
```

以下是回應範例。

```
{
```

```

    "Activities": [
      {
        "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",
        "AutoScalingGroupName": "my-asg",
        "Description": "Moving EC2 instance out of Standby:
i-05b4f7d5be44822a6",
        "Cause": "At 2023-12-15T21:46:14Z instance i-05b4f7d5be44822a6 was
moved out of standby in
        response to a user request, increasing the capacity from 3 to
4.",
        "StartTime": "2023-12-15T21:46:14.678Z",
        "StatusCode": "PreInService",
        "Progress": 30,
        "Details": "{\"Subnet ID\": \"subnet-c934b782\", \"Availability Zone
\": \"us-west-2a\"}"
      }
    ]
  }
}

```

6. (選用) 使用以下 `describe-auto-scaling-instances` 命令驗證恢復到服務狀態的執行個體。

```
aws autoscaling describe-auto-scaling-instances --instance-ids i-05b4f7d5be44822a6
```

以下是回應範例。請注意，該執行個體的狀態為 `InService`。

```

{
  "AutoScalingInstances": [
    {
      "ProtectedFromScaleIn": false,
      "AvailabilityZone": "us-west-2a",
      "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
      },
      "InstanceId": "i-05b4f7d5be44822a6",
      "InstanceType": "t3.micro",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService"
    },
  ]
}

```

```
    ...  
  ]  
}
```

刪除 Auto Scaling 基礎設施

若要完全刪除您的擴展基礎設施，請完成以下任務。

任務

- [刪除 Auto Scaling 群組](#)
- [\(選用\) 刪除啟動組態](#)
- [\(選用\) 刪除啟動範本](#)
- [\(選用\) 刪除負載平衡器和目標群組](#)
- [\(選用\) 刪除 CloudWatch 警示](#)
- [設定 Amazon EC2 Auto Scaling 資源的刪除保護](#)

刪除 Auto Scaling 群組

刪除 Auto Scaling 群組時，其所需的最小值和最大值會設定為 0。因此，執行個體會終止。刪除執行個體也會刪除任何關聯的日誌或資料，以及執行個體上的任何磁碟區。如果不要終止一或多個執行個體，您可以將它們分離，然後再刪除 Auto Scaling 群組。如果群組具有擴展政策，則刪除群組會刪除政策、基礎警示動作，以及任何不再具有相關聯動作的警示。

若要刪除 Auto Scaling 群組 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁邊的核取方塊，然後選擇動作、刪除。
3. 出現確認提示時，請輸入 **delete** 來確認刪除特定的 Auto Scaling 群組，然後選擇 Delete (刪除)。

Name (名稱) 欄位中的載入圖示會顯示正在刪除 Auto Scaling 群組。Desired (所需)、Min (最小) 和 Max (最大) 資料欄會顯示 Auto Scaling 群組的 0 個執行個體。終止執行個體並刪除群組需要幾分鐘的時間。重新整理清單，以查看目前狀態。

若要刪除 Auto Scaling 群組 (AWS CLI)

使用以下 [delete-auto-scaling-group](#) 命令來刪除 Auto Scaling 群組。此操作並不能用於擁有 EC2 執行個體的群組，而只能用於沒有任何執行個體的群組。

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg
```

如果群組有執行個體或進行中的擴展活動，請將 [delete-auto-scaling-group](#) 命令與 `--force-delete` 選項搭配使用。這也會終止 EC2 執行個體。當您從 Amazon EC2 Auto Scaling 主控台刪除 Auto Scaling 群組，主控台就會使用此操作來同時終止任何 EC2 執行個體和刪除群組。

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg --force-delete
```

(選用) 刪除啟動組態

如果您想要保留啟動組態以供日後使用，則可略過此步驟。

刪除啟動組態 (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在左側導覽窗格的 Auto Scaling 下，選擇 Auto Scaling 群組。
3. 在頁面頂端附近選擇啟動組態。出現確認提示時，請選擇檢視啟動組態，以確認您要檢視啟動組態頁面。
4. 選取啟動組態，然後選擇動作、複製啟動組態。
5. 出現確認提示時，請選擇刪除。

刪除啟動組態 (AWS CLI)

使用以下 [delete-launch-configuration](#) 命令：

```
aws autoscaling delete-launch-configuration --launch-configuration-name my-launch-config
```

(選用) 刪除啟動範本

您可以刪除啟動範本或只刪除某個版本的啟動範本。刪除啟動範本時，會刪除其所有版本。

您可以略過此步驟，來保留啟動範本以供日後使用。

刪除您的啟動範本 (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在導覽窗格的 Instances (執行個體) 底下，選擇 Launch Templates (啟動範本)。
3. 選擇您的啟動範本，然後執行下列其中一項：
 - 選擇 Actions (動作)、Delete template (刪除範本)。出現確認提示時，請輸入 **Delete** 確認刪除指定的啟動範本，然後選擇 Delete (刪除)。
 - 選擇 Actions (動作)、Delete template version (刪除範本版本)。選取要刪除的版本，然後選取 Delete (刪除)。

刪除啟動範本 (AWS CLI)

請使用下列 [delete-launch-template](#) 命令來刪除您的範本及其所有版本。

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b72934aff71
```

您也可以使用 [delete-launch-template-versions](#) 命令來刪除特定版本的啟動範本。

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b72934aff71 --versions 1
```

(選用) 刪除負載平衡器和目標群組

如果您的 Auto Scaling 群組未與 Elastic Load Balancing 負載平衡器相關聯，或您想要保留此負載平衡器供日後使用，請略過此步驟。

刪除負載平衡器 (主控台)

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在導覽窗格的 Load Balancing (負載平衡器)，選擇 Load Balancer (負載平衡器)。
3. 選擇負載平衡器，然後選擇 Actions (動作)、Delete (刪除)。
4. 出現確認提示時，選擇 Yes, Delete (是，刪除)。

刪除目標群組 (主控台)

1. 在導覽窗格的 Load Balancing (負載平衡) 中，選擇 Target Groups (目標群組)。

2. 選取目標群組，然後依序選擇 Actions (動作)、Delete (刪除)。
3. 出現確認提示時，選擇 Yes, Delete (是，刪除)。

刪除與 Auto Scaling 群組關聯的負載平衡器 (AWS CLI)

針對 Application Load Balancer 和網路負載平衡器，請使用下列 [delete-load-balancer](#) 和 [delete-target-group](#) 命令。

```
aws elbv2 delete-load-balancer --load-balancer-arn my-load-balancer-arn  
aws elbv2 delete-target-group --target-group-arn my-target-group-arn
```

針對 Classic Load Balancer，請使用下列 [delete-load-balancer](#) 命令。

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

(選用) 刪除 CloudWatch 警示

若要刪除與 Auto Scaling 群組相關聯的 CloudWatch 警示，請完成下列步驟。例如，您可能會有與步進擴展或簡易擴展政策相關聯的警示。

Note

刪除 Auto Scaling 群組會自動刪除 Amazon EC2 Auto Scaling 為目標追蹤擴展政策管理的 CloudWatch 警示。

如果您的 Auto Scaling 群組未與任何 CloudWatch 警示建立關聯，或您想要保留此警示以供日後使用，則可略過此步驟。

刪除 CloudWatch 警示 (主控台)

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格上，選擇 Alarms (警示)。
3. 選擇警示，然後依序選擇 Action (動作)、Delete (刪除)。
4. 出現確認提示時，請選擇刪除。

刪除 CloudWatch 警示 (AWS CLI)

使用 [delete-alarms](#) 命令。您可以同時刪除一或多個警示。例如，使用下列命令來刪除 Step-Scaling-AlarmHigh-AddCapacity 和 Step-Scaling-AlarmLow-RemoveCapacity 警示。

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-RemoveCapacity
```

設定 Amazon EC2 Auto Scaling 資源的刪除保護

透過設定多層保護，保護您的 Amazon EC2 Auto Scaling 基礎設施免於意外刪除。Auto Scaling 提供多種方法來防止您 Auto Scaling 群組及其管理的 Amazon EC2 執行個體遭到不必要的資源刪除。

目錄

- [設定 Auto Scaling 群組刪除保護](#)
- [使用 IAM 政策控制刪除許可](#)

設定 Auto Scaling 群組刪除保護

刪除保護是一種資源層級設定，可防止 Amazon EC2 Auto Scaling 群組意外刪除。啟用時，刪除保護會阻止 [DeleteAutoScalingGroup](#) API 操作成功，要求您先將刪除保護設定更新為較不嚴格的層級，才能刪除 Auto Scaling 群組。

Amazon EC2 Auto Scaling 提供三種層級的刪除保護：

無（預設）

未啟用刪除保護，這表示您的 Auto Scaling 群組可以使用或不使用 ForceDelete 選項來刪除。使用 ForceDelete 時，由 Auto Scaling 群組管理的所有 Amazon EC2 執行個體也會強制終止，而不會執行終止生命週期關聯。

防止強制刪除

使用 ForceDelete 選項時，無法刪除 Auto Scaling 群組。此組態允許刪除空的 Auto Scaling 群組（沒有執行個體的群組）。對於您想要防止大量執行個體終止，但允許清除空白群組的生產工作負載，建議使用此選項。

防止所有刪除

無論是否使用 ForceDelete 選項，都無法刪除 Auto Scaling 群組。此選項提供最強大的保護，防止意外刪除。它需要明確停用刪除保護，才能刪除 Auto Scaling 群組。對於應該很少刪除或永遠不會刪除的任務關鍵 Auto Scaling 群組，建議使用此選項。

刪除保護的運作方式

當您嘗試啟用刪除保護的 [DeleteAutoScalingGroup](#) API 操作時：

1. Amazon EC2 Auto Scaling 會在處理請求之前驗證刪除保護設定。
2. 如果設定的刪除保護層級封鎖刪除嘗試，Amazon EC2 Auto Scaling 會傳回 `ValidationError`。
3. 您的 Auto Scaling 群組及其 Amazon EC2 執行個體保持不變。
4. 您必須先將刪除保護設定更新為較不嚴格的層級，才能刪除 Auto Scaling 群組。

刪除保護不會阻止其他操作，例如：

- 更新 Auto Scaling 群組組態。
- 終止個別執行個體。
- 擴展操作（手動或自動）。
- 暫停或繼續程序。

如需如何正常處理執行個體終止的詳細資訊，請參閱 [設計您的應用程式以正常處理執行個體終止](#)。

設定刪除保護

您可以在建立 Auto Scaling 群組或更新現有 Auto Scaling 群組的設定時設定刪除保護。

Console

建立具有刪除保護的 Auto Scaling 群組

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。
3. 完成 Auto Scaling 群組的組態步驟。
4. 在設定群組大小和擴展頁面上，展開其他設定。
5. 對於 Auto Scaling 群組刪除保護，請選擇所需的保護層級：
 - 無 - 無刪除保護（預設）
 - 防止強制刪除 - 封鎖強制刪除操作
 - 防止所有刪除 - 封鎖所有刪除操作

6. 完成剩餘的步驟以建立 Auto Scaling 群組。

更新現有 Auto Scaling 群組的刪除保護

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。
3. 選擇 Actions (動作)、Edit (編輯)。
4. 在其他設定下，更新 Auto Scaling 群組刪除保護設定。
5. 選擇更新。

AWS CLI

建立具有刪除保護的 Auto Scaling 群組

使用 [create-auto-scaling-group](#) 命令搭配 `--deletion-protection` 參數：

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateName=my-template,Version='$Latest' \  
  --min-size 1 \  
  --max-size 5 \  
  --desired-capacity 2 \  
  --vpc-zone-identifier "subnet-12345678,subnet-87654321" \  
  --deletion-protection prevent-force-deletion
```

的有效值 `--deletion-protection` 為：`none` | `prevent-force-deletion` | `prevent-all-deletion`

更新現有 Auto Scaling 群組的刪除保護

使用 [update-auto-scaling-group](#) 命令：

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --deletion-protection prevent-all-deletion
```

停用刪除保護

將刪除保護設定為 `none`：

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --deletion-protection none
```

驗證刪除保護狀態

使用 [describe-auto-scaling-groups](#) 命令：

```
aws autoscaling describe-auto-scaling-groups \  
  --auto-scaling-group-names my-asg
```

使用 IAM 政策控制刪除許可

使用 AWS Identity and Access Management (IAM) 政策來控制哪些使用者和角色可以刪除 Auto Scaling 群組。IAM 型控制項透過限制身分層級的許可，提供額外的安全層。

當您想要執行下列動作時，IAM 政策特別有用：

- 允許不同使用者存取不同層級的 Auto Scaling 操作。
- 防止特定使用者使用 `ForceDelete` 選項，即使他們可以執行其他 Auto Scaling 操作。
- 限制特定 Auto Scaling 群組的刪除許可。

只有在群組有標籤 **`environment=development`** 時，下列政策才允許刪除 Auto Scaling 群組。

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [{  
    "Effect": "Allow",  
    "Action": "autoscaling:DeleteAutoScalingGroup",  
    "Resource": "*",  
    "Condition": {  
      "StringEquals": { "aws:ResourceTag/environment": "development" }  
    }  
  }  
}]
```

```
}
```

下列政策使用 `autoscaling:ForceDelete` 條件金鑰來控制對 `DeleteAutoScalingGroup` API 動作的存取。這可防止某些使用者使用 `ForceDelete` 操作，這會終止 Auto Scaling 群組中的所有 Amazon EC2 執行個體。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": "autoscaling:DeleteAutoScalingGroup",
    "Resource": "*",
    "Condition": {
      "Bool": {
        "autoscaling:ForceDelete": "true"
      }
    }
  }]
}
```

或者，如果您不使用條件金鑰來控制 Auto Scaling 群組的存取權，則可以指定 `Resource` 元素中資源的 ARN 來控制存取權。

下列政策提供使用者使用 `DeleteAutoScalingGroup` API 動作的許可，但僅適用於名稱開頭為 `devteam-` 的 Auto Scaling 群組。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "arn:aws:autoscaling:us-east-1:111122223333:autoScalingGroup:*:autoScalingGroupName/devteam-*"
    }
  ]
}
```

```
]
}
```

您也可以將多個 ARN 包含在清單中來加以指定。包含 UUID 以確保將存取權授予特定 Auto Scaling 群組。新群組的 UUID 與同名之已刪除群組的 UUID 不同。

```
"Resource": [
  "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-1",
  "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-2",
  "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-3"
]
```

如需 Amazon EC2 Auto Scaling IAM 政策的其他範例，包括控制刪除許可的政策，請參閱 [身分型政策範例](#)。

取代 Auto Scaling 群組中的執行個體

Amazon EC2 Auto Scaling 提供的功能可讓您在更新後取代 Auto Scaling 群組中的 Amazon EC2 執行個體，例如使用新的 Amazon Machine Image (AMI) 新增新的啟動範本，或新增新的執行個體類型。它還可讓您選擇將更新包含在取代執行個體的相同操作中，以協助您簡化更新。

本節包含協助您執行下列操作的資訊：

- 啟動執行個體重新整理來取代 Auto Scaling 群組中的執行個體。
- 宣告描述所需組態的特定更新，並將 Auto Scaling 群組更新為所需組態。
- 略過取代已更新的執行個體。
- 使用檢查點分階段更新執行個體，並在特定點對執行個體執行驗證。
- 使用製作時間在執行個體重新整理結束時暫停，以驗證執行個體的運作狀態。
- 到達檢查點時，透過電子郵件接收通知。
- 使用復原將 Auto Scaling 群組還原為先前使用的組態。
- 如果執行個體重新整理因某些原因失敗，或者您指定的任何 Amazon CloudWatch 警示進入 ALARM 狀態，則會自動復原。
- 限制執行個體的生命週期，以在整個 Auto Scaling 群組中提供一致的軟體版本和執行個體組態。
- 在不終止執行個體的情況下取代根磁碟區，同時保留網路介面、非根磁碟區和 IAM 政策。

目錄

- [使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)
- [根據執行個體生命週期上限來取代 Auto Scaling 執行個體](#)

使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體

您可以使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體。當組態變更需要您取代執行個體或其根磁碟區時，此功能非常有用，特別是當您的 Auto Scaling 群組包含大量執行個體時。

執行個體重新整理的某些情況有助於包括：

- 在 Auto Scaling 群組中部署新的 Amazon Machine Image (AMI) 或使用者資料指令碼。您可以使用變更建立新的啟動範本，然後使用執行個體重新整理來立即推出更新。
- 將執行個體遷移至新的執行個體類型，以利用最新的改善和最佳化。

- 將 Auto Scaling 群組從使用啟動組態切換到使用啟動範本。您可以複製啟動組態以啟動範本，然後使用執行個體重新整理將執行個體更新為新的範本。如需遷移至啟動範本的詳細資訊，請參閱 [遷移 Auto Scaling 群組以啟動範本](#)。
- 套用安全修補程式或軟體更新，同時保留長時間執行的執行個體狀態，並避免使用 GPU 或 Mac 執行個體等特殊執行個體類型的容量限制。

目錄

- [執行個體重新整理如何在 Auto Scaling 群組中運作](#)
- [了解執行個體重新整理的預設值](#)
- [使用 AWS 管理主控台 或 啟動執行個體重新整理 AWS CLI](#)
- [使用 AWS 管理主控台 或 監控執行個體重新整理 AWS CLI](#)
- [在執行個體重新整理期間取代根磁碟區](#)
- [使用 AWS 管理主控台 或 取消執行個體重新整理 AWS CLI](#)
- [使用手動或自動轉返復原變更](#)
- [搭配略過相符項目使用執行個體重新整理](#)
- [將檢查點新增至執行個體重新整理](#)

執行個體重新整理如何在 Auto Scaling 群組中運作

本主題說明執行個體重新整理的運作方式，並介紹有效使用它所需的重要概念。

目錄

- [運作方式](#)
- [核心概念](#)
- [運作狀態檢查寬限期](#)
- [執行個體類型相容性](#)
- [限制](#)

運作方式

若要重新整理 Auto Scaling 群組中的執行個體，您可以定義新的組態，其中包含應用程式的最新版本，以及您想要進行的任何其他更新。

執行個體重新整理支援兩種更新執行個體的策略：

- 滾動策略（預設）— 根據您的偏好設定終止執行個體並批次啟動新的執行個體。這可確保 Auto Scaling 群組在整個更新過程中維持所需的容量和可用性。
- 取代根磁碟區策略 — 僅取代根磁碟區來更新執行個體，而不終止執行個體。這會保留執行個體網路介面、非根 EBS 磁碟區和執行個體存放區資料。

取代根磁碟區策略的需求：

- Auto Scaling 群組必須使用混合執行個體政策
- 混合執行個體政策中的所有覆寫都必須指定 ImageId
- AMIs 只能包含單一根磁碟區
- 所有執行個體都必須符合群組的啟動範本組態
- 您必須使用具有包含 ImageId 覆寫之混合執行個體政策的所需組態來啟動執行個體重新整理。

啟用略過比對時，Auto Scaling 會將每個執行個體目前的 AMI ID 與您所需組態中的 AMI IDs 進行比較。它只會取代 AMI IDs 不相符的執行個體，讓您略過已更新的執行個體。

執行執行個體重新整理

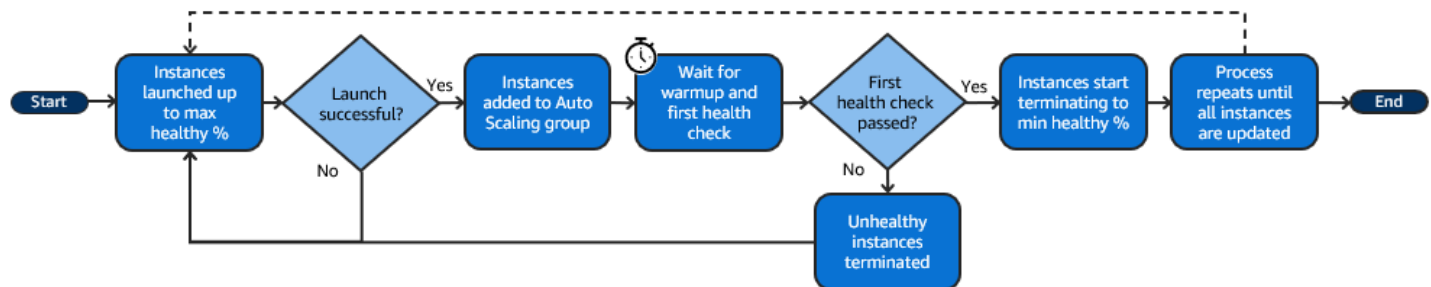
啟動執行個體重新整理，以根據該組態將現有執行個體取代為新的執行個體。

1. 建立新的啟動範本，或使用所需的組態變更更新現有範本，例如新的 Amazon Machine Image (AMI)。如需詳細資訊，請參閱 [為 Auto Scaling 群組建立啟動範本](#)。
2. 使用 Amazon EC2 Auto Scaling 主控台 AWS CLI 或 SDK 啟動執行個體重新整理：
 - 指定您建立的新啟動範本或啟動範本版本。這將用於啟動新的執行個體。
 - 設定偏好的運作狀態百分比下限和上限。這可控制同時取代多少執行個體，以及在終止舊執行個體之前是否啟動新執行個體。
 - 設定任何選用設定，例如：
 - 檢查點 – 在特定百分比的取代之後暫停執行個體重新整理，以驗證進度。
 - 製作時間 – 在執行個體重新整理結束時暫停，以在執行個體重新整理視為完成之前驗證執行個體的運作狀態。
 - 略過比對 – 比較舊執行個體與新組態，並僅取代不相符的執行個體。當您從主控台啟動執行個體重新整理時，略過比對預設為開啟。
 - 多個執行個體類型 – 將新的或更新的 [混合執行個體政策](#) 套用為所需組態的一部分。

當執行個體重新整理開始時，Amazon EC2 Auto Scaling 將：

- 根據運作狀態百分比下限和上限，批次取代執行個體。
- 如果運作狀態最低百分比設定為 100%，請先啟動新執行個體，再終止舊執行個體。這可確保隨時維持所需的容量。
- 檢查執行個體的運作狀態，並在取代更多執行個體之前給予他們暖機時間。
- 終止並取代發現運作狀態不佳的執行個體。
- 在執行個體重新整理成功後，使用新的組態變更自動更新 Auto Scaling 群組設定。
- 在暖集區中的InService執行個體之前取代執行個體。

下列流程圖說明當您將運作狀態最低百分比設定為 100% 時，終止行為之前的啟動。



Note

只有在您未設定執行個體維護政策，或需要覆寫現有政策時，才需要指定執行個體重新整理的運作狀態百分比下限和上限。如需詳細資訊，請參閱[執行個體維護政策](#)。

同樣地，如果您尚未啟用預設暖機期，或需要覆寫預設值，則只需要指定執行個體重新整理的執行個體暖機期。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

核心概念

在開始使用之前，請熟悉以下執行個體重新整理核心概念：

狀態良好最低百分比

運作狀態良好的最低百分比是在執行個體重新整理期間保持服務狀態、運作狀態良好且可供使用的所需容量百分比，以便繼續重新整理。例如，如果狀態良好最低百分比為 90%，而狀態良好最高百分比為 100%，則一次會取代 10% 的容量。如果新執行個體未通過運作狀態檢查，Amazon EC2 Auto Scaling 會終止並取代執行個體。如果執行個體重新整理無法啟動任何運作狀態良好的執行個

體，則執行個體重新整理最終會失敗，而不會影響其他 90% 的群組。如果新執行個體保持運作狀態並完成暖機期，Amazon EC2 Auto Scaling 可以繼續取代其他執行個體。

執行個體重新整理可以一次取代一個執行個體、一次取代多個執行個體，或一次取代所有執行個體。若要一次取代一個執行個體，請將狀態良好最低和最高百分比均設定為 100%。這會將執行個體重新整理的行為變更為在終止前啟動，以防止群組的容量降低至其所需容量的 100% 以下。若要一次取代所有執行個體，請將狀態良好最低百分比設為 0%。

狀態良好最高百分比

狀態良好最高百分比是 Auto Scaling 群組在取代執行個體時可增加的所需容量百分比。上下限之間的差異不能大於 100。範圍越大，可同時取代的執行個體數量便會越多。

執行個體暖機期

執行個體暖機期是指新執行個體的状态變更為 InService 到視為已完成初始化經過的時間。在執行個體重新整理期間，如果執行個體通過其運作狀態檢查，Amazon EC2 Auto Scaling 在判斷新啟動的執行個體運作狀態良好之後，就不會立即繼續取代下一個執行個體。它會等待暖機期，然後再繼續取代下一個執行個體。如果您的應用程式仍需要一些初始化時間才能回應請求，這個功能會相當實用。

執行個體暖機期的運作方式與預設執行個體暖機期相同。因此適用相同的擴展考量。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

所需組態

所需組態是您希望 Amazon EC2 Auto Scaling 在整個 Auto Scaling 群組中部署的新組態。例如，您可為執行個體指定新啟動範本和新執行個體類型。在執行個體重新整理期間，Amazon EC2 Auto Scaling 會將 Auto Scaling 群組更新為所需組態。如果在執行個體重新整理期間發生擴增事件，Amazon EC2 Auto Scaling 會以所需組態 (而不是群組的目前設定) 啟動新執行個體。執行個體重新整理成功後，Amazon EC2 Auto Scaling 會更新 Auto Scaling 群組設定，以便反映您在執行個體重新整理過程中指定的所需新組態。

略過相符項目

略過相符項目會讓 Amazon EC2 Auto Scaling 忽略已經有最新更新的執行個體。如此一來，您就不會取代超過需要的執行個體數量。當您想要確定 Auto Scaling 群組使用特定版本的啟動範本，並且僅取代使用不同版本的執行個體時，這會很有幫助。

檢查點

檢查點是執行個體重新整理暫停一段指定時間的時間點。執行個體重新整理可以包含多個檢查點。Amazon EC2 Auto Scaling 會為每個檢查點發出事件。因此，您可新增 EventBridge 規則以將

事件傳送到某個目標 (例如 Amazon SNS)，以便在到達檢查點時收到通知。到達檢查點後，您就能夠驗證部署。如果發現任何問題，您可以取消執行個體重新整理或進行回復。分階段部署更新的能力是檢查點的主要優點。如果不使用檢查點，輪流取代將持續執行。

若要深入了解啟動執行個體重新整理時可設定的所有預設設定，請參閱 [了解執行個體重新整理的預設值](#)。

運作狀態檢查寬限期

Amazon EC2 Auto Scaling 會根據您的 Auto Scaling 群組使用的運作狀態檢查狀態，判斷執行個體是否運作狀態良好。如需詳細資訊，請參閱 [Auto Scaling 群組中的執行個體運作狀態檢查](#)。

為確保這些運作狀態檢查能夠盡快啟動，請勿將群組的運作狀態檢查寬限期設定得太高，但要足以讓您的 Elastic Load Balancing 運作狀態檢查來判斷目標是否可用來處理要求。如需詳細資訊，請參閱 [設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。

執行個體類型相容性

在變更執行個體類型之前，最好先確認它可以與您的啟動範本搭配運作。這會確認與您指定的 AMI 的相容性。例如，假設您已從全虛擬化 (PV) AMI 啟動原始執行個體，但想要變更為只有硬體虛擬機器 (HVM) AMI 支援的目前世代執行個體類型。在這種情況下，您必須在啟動範本中使用 HVM AMI。

若要在不啟動執行個體的情況下確認執行個體類型相容性，請使用 [run-instances](#) 命令搭配 `--dry-run` 選項，如下列範例所示。

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --dry-run
```

如需如何判斷相容性的詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [變更執行個體類型的相容性](#)。

限制

- 總持續時間 — 執行個體重新整理可繼續主動取代執行個體的時間上限為 14 天。
- 加權群組特定行為的差異：如果混合執行個體群組的執行個體權重設定為大於或等於群組所需容量，Amazon EC2 Auto Scaling 可能會一次取代所有 InService 執行個體。若要避免這種情況，請遵循 [設定 Auto Scaling 群組以使用執行個體權重](#) 主題中的建議。當您將權重與 Auto Scaling 群組搭配使用時，請指定大於您最大權重的所需容量。

- 一小時逾時 - 當執行個體重新整理因為等待取代待命或受到縮減保護的執行個體，或新執行個體未通過運作狀態檢查，而無法繼續進行取代時，Amazon EC2 Auto Scaling 會繼續重試一小時。它也會提供狀態訊息，以協助您解決問題。如果問題在一小時後仍然存在，操作會失敗。其目的是在發生暫時性問題時，給它時間恢復。
- 透過使用者資料部署程式碼 — 略過比對不會檢查從使用者資料指令碼部署的程式碼變更。如果您使用使用者資料提取新程式碼並在新執行個體上安裝這些更新，我們建議您關閉略過比對，以確保所有執行個體都收到最新的程式碼，即使沒有啟動範本版本更新也一樣。
- 更新限制 — 如果您在具有所需組態的執行個體重新整理處於作用中狀態時嘗試更新 Auto Scaling 群組的啟動範本、啟動組態或混合執行個體政策，請求將失敗，並出現下列驗證錯誤：An active instance refresh with a desired configuration exists. All configuration options derived from the desired configuration are not available for update while the instance refresh is active.
- 以屬性為基礎的執行個體選擇 — 如果您的 Auto Scaling 群組使用以屬性為基礎的執行個體選擇（在混合執行個體政策InstanceRequirements中指定），則不支援下列執行個體重新整理參數：
 - SkipMatching - 如果您使用已InstanceRequirements設定的 Auto Scaling 群組中的 SkipMatching 參數啟動執行個體重新整理，則會失敗。
 - DesiredConfiguration - 如果您使用已InstanceRequirements設定的 Auto Scaling 群組中的 DesiredConfiguration 參數啟動執行個體重新整理，則會失敗。

如果您需要在具有屬性型執行個體選擇的 Auto Scaling 群組上執行執行個體重新整理，請在沒有這些參數的情況下啟動執行個體重新整理。

- 根磁碟區取代不支援 Elastic Load Balancing。

了解執行個體重新整理的預設值

啟動執行個體重新整理之前，您可以先自訂影響執行個體重新整理的各種偏好設定。根據您是否使用主控台或命令列 (AWS CLI 或 AWS SDK)，某些偏好設定預設值會有所不同。

下表列出執行個體重新整理設定的預設值。

設定	AWS CLI 或 AWS SDK	Amazon EC2 Auto Scaling 主控台
CloudWatch 警示	已停用 (null)	Disabled
自動復原	已停用 (false)	Disabled

設定	AWS CLI 或 AWS SDK	Amazon EC2 Auto Scaling 主控台
製作時間	零	零
檢查點	已停用 (false)	Disabled
檢查點延遲	1 小時 (3600 秒)	1 小時
執行個體暖機期	預設執行個體暖機期 (如果已定義) 或 運作狀態檢查寬限期 。	預設執行個體暖機期 (如果已定義) 或 運作狀態檢查寬限期 。
狀態良好最高百分比	根據您的執行個體維護政策而異。如果沒有執行個體維護政策，此預設值為 100% (null)。	根據您的執行個體維護政策而異。如果沒有執行個體維護政策，此預設值為 100% (null)。
狀態良好最低百分比	根據您的執行個體維護政策而異。如果沒有執行個體維護政策，此預設值為 90%。	根據您的執行個體維護政策而異。如果沒有執行個體維護政策，此預設值為 90%。
受縮減保護的執行個體	等候	Ignore
略過相符項目	已停用 (false)	已啟用
待命執行個體	等候	Ignore

各項設定的描述如下：

CloudWatch 警示 (AlarmSpecification)

CloudWatch 警示規格。CloudWatch 警示可用於識別任何問題，並在警示進入 ALARM 狀態時使作業失敗。如需詳細資訊，請參閱[使用自動復原啟動執行個體重新整理](#)。

自動復原 (AutoRollback)

控制在執行個體重新整理失敗的情況下，Amazon EC2 Auto Scaling 是否要將 Auto Scaling 群組復原為先前的組態。如需詳細資訊，請參閱[使用手動或自動轉返復原變更](#)。

製作時間 (BakeTime)

在執行個體重新整理結束時，執行個體重新整理視為完成之前等待的時間量。

檢查點 (CheckpointPercentages)

控制 Amazon EC2 Auto Scaling 是否分階段取代執行個體。如果您需要在取代所有執行個體之前對執行個體執行驗證，這將非常有用。如需詳細資訊，請參閱[將檢查點新增至執行個體重新整理](#)。

檢查點延遲 (CheckpointDelay)

到檢查點後再繼續前的等待時間 (以秒為單位)。如需詳細資訊，請參閱[將檢查點新增至執行個體重新整理](#)。

執行個體暖機期 (InstanceWarmup)

一段單位為秒的時間期間，是指 Amazon EC2 Auto Scaling 等到新執行個體被視為已完成初始化後再繼續取代下一個執行個體所經過的時間。如果您已經正確定義 Auto Scaling 群組的預設執行個體暖機期，就不需要變更執行個體暖機期 (除非您想要覆寫預設值)。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

狀態良好最高百分比 (MaxHealthyPercentage)

Auto Scaling 群組在取代執行個體時可增加的所需容量百分比。

狀態良好百分比下限 (MinHealthyPercentage)

Auto Scaling 群組得以正常運作、保持狀態良好、以及隨時可用以便繼續操作的所需容量百分比。

受縮減保護的執行個體 (ScaleInProtectedInstances)

控制 Amazon EC2 Auto Scaling 在找到受縮減保護的執行個體時所執行的動作。如需這些執行個體的詳細資訊，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。

Amazon EC2 Auto Scaling 提供下列選項：

- 取代 (Refresh) – 取代受保護免於縮減的執行個體。
- 忽略 (Ignore) – 忽略受保護免於縮減的執行個體，並繼續取代未受保護的執行個體。
- 等待 (Wait) – 等待一小時讓您移除縮減保護。如果不這樣做，執行個體重新整理就會失敗。

略過相符項目 (SkipMatching)

控制 Amazon EC2 Auto Scaling 是否略過取代符合所需組態的執行個體。如果未指定所需組態，則會略過取代具有與 Auto Scaling 群組在執行個體重新整理開始前所使用之相同啟動範本和執行個體類型的執行個體。如需詳細資訊，請參閱[搭配略過相符項目使用執行個體重新整理](#)。

待命執行個體 (StandbyInstances)

控制 Amazon EC2 Auto Scaling 在找到處於 Standby 狀態的執行個體時所執行的動作。如需這些執行個體的詳細資訊，請參閱[從 Auto Scaling 群組暫時移除執行個體](#)。

Amazon EC2 Auto Scaling 提供下列選項：

- 終止 (Terminate) – 終止 中的執行個體Standby。
- 忽略 (Ignore) – 忽略 中的執行個體，Standby並繼續取代 狀態的執行個體InService。
- 等待 (Wait) – 等待一小時，讓您將執行個體送回 服務。如果不這樣做，執行個體重新整理就會失敗。

使用 AWS 管理主控台 或 啟動執行個體重新整理 AWS CLI

Important

您可復原進行中的執行個體重新整理以復原任何變更。若要使用此功能，Auto Scaling 群組必須符合在開始執行個體重新整理前使用復原的先決條件。如需詳細資訊，請參閱[使用手動或自動轉返復原變更](#)。

下列程序可協助您使用 AWS 管理主控台 或 啟動執行個體重新整理 AWS CLI。

啟動執行個體重新整理 (主控台)

如果您是第一次啟動執行個體重新整理，則使用主控台將有助於您了解這些可用的功能和選項。

在主控台中啟動執行個體重新整理 (基本程序)

如果您之前未定義 Auto Scaling 群組的[混合執行個體政策](#)，請使用下列程序。如果您之前定義了混合執行個體政策，請參閱 [在主控台中啟動執行個體重新整理 \(混合執行個體群組\)](#) 以啟動執行個體重新整理。

啟動執行個體重新整理

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Instance refresh (執行個體重新整理) 索引標籤的 Instance refreshes (執行個體重新整理) 中，選擇 Start instance refresh (啟動執行個體重新整理)。
4. 針對可用性設定，執行以下操作：

a. 針對執行個體取代方法：

- 如果您尚未在 Auto Scaling 群組上設定執行個體維護政策，則執行個體取代方法的預設設定為終止並啟動。這是執行個體重新整理的舊版預設行為。
- 如果您在 Auto Scaling 群組上設定了執行個體維護政策，則該政策會提供執行個體取代方法的預設值。若要覆寫執行個體維護政策，請選擇覆寫。覆寫功能僅適用於目前的執行個體重新整理。下次啟動執行個體重新整理時，這些值會重設為執行個體維護政策的預設值。

下列程序說明如何更新執行個體取代方法。

i. 選擇下列其中一種執行個體取代方法：

- 終止前啟動：必須先佈建新執行個體，才能終止現有執行個體。對於重視可用性多於節省成本的應用程式來說，這是理想的選擇。
- 終止並啟動：在終止現有執行個體的同時佈建新執行個體。對於重視節省成本多於可用性的應用程式來說，這是理想的選擇。對於不應啟動超過目前可用容量的應用程式而言，這也是一個不錯的選擇。
- 自訂行為：此選項可讓您根據想要在取代執行個體時的可用容量來設定自訂上下限範圍。這可協助您在成本和可用性之間取得適當的平衡。

ii. 針對設定正常運作狀態百分比，請輸入下列一個或兩個欄位的值。啟用欄位會因您為執行個體取代方法所選的選項而異。

- 下限：設定繼續執行個體重新整理所需的狀態良好最低百分比。
- 上限：設定執行個體重新整理時可能的狀態良好最高百分比。

iii. 展開根據目前群組大小在取代期間檢視預估的臨時容量區段，以確認如何在群組中套用下限和上限。使用的實際值取決於所需的容量值，此值會隨群組擴展而有所改變。

iv. 展開設定無效取代大小的回復行為區段，然後選擇是否違反正常運作狀態百分比上限，以排列可用性的優先順序，或選擇違反正常運作狀態百分比下限。

對於非常小的群組，不建議保留預設的違反正常運作狀態百分比下限選項。如果 Auto Scaling 群組中只有一個執行個體，啟動執行個體重新整理可能會導致中斷。

如果您使用的 Auto Scaling 群組尚未設定執行個體維護政策，則此步驟會設定回復行為。當您的群組有執行個體維護政策時，此選項無法使用，也不會顯示。另外，此選項僅適用於終止並啟動取代方法。其他取代方法將違反狀態良好最高百分比，以便優先考慮可用性。

- b. 在執行個體暖機期中，輸入從新執行個體的狀態變更為 InService 到完成初始化所經過的秒數。Amazon EC2 Auto Scaling 會等待這段時間，然後再繼續取代下一個執行個體。

暖機時，新啟動的執行個體也不會計入 Auto Scaling 群組的彙總執行個體指標 (例如 CPUUtilization、NetworkIn 和 NetworkOut)。如果將擴展政策新增至 Auto Scaling 群組，則擴展活動會平行執行。如果您為執行個體重新整理暖機期設定了長間隔，則新啟動的執行個體在指標中顯示需要更多時間。因此，足夠的暖機期可防止 Amazon EC2 Auto Scaling 擴展過時的指標資料。

如果您已經正確定義 Auto Scaling 群組的預設執行個體暖機期，就不需要變更執行個體暖機期。但是，如果您想要覆寫預設值，您可以為此選項設定一個值。如需有關設定預設執行個體暖機期的詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

5. 針對重新整理設定，執行以下操作：

- a. (選用) 在 Checkpoints (檢查點) 欄位中，選擇 Enable checkpoints (啟用檢查點)，以使用對執行個體重新整理的增量或分階段方法來取代執行個體。這會為每次取代之間的驗證提供額外時間。如果您選擇不啟用檢查點，則執行個體會在一次近乎連續的操作中被取代。

如果您要啟用檢查點，請參閱[啟用檢查點 \(主控台\)](#) 了解額外的步驟。

- b. (選用) 對於製作時間，指定執行個體重新整理結束時，執行個體重新整理視為完成之前要等待的時間量。
- c. 啟用或關閉 Skip matching (略過相符項目)：
 - 若要略過取代已與啟動範本相符的執行個體，請讓啟用略過相符項目核取方塊保持為已勾選的狀態。
 - 如果清除此核取方塊來關閉略過相符項目，則會取代所有執行個體。

當您啟用跳過比對時，您可以設定新啟動範本或啟動範本的新版本，而不是使用現有啟動範本。可以在啟動執行個體重新整理頁面的所需組態區段中執行這項操作。

Note

若要使用略過相符項目功能來更新目前使用啟動組態的 Auto Scaling 群組，您必須在 Desired configuration (所需設定) 中選擇啟動範本。不支援搭配啟動組態執行跳過比對動作。

- d. 對於待命執行個體，請選擇忽略、終止或等待。這會決定找到 Standby 狀態的執行個體時所發生的情況。如需詳細資訊，請參閱[從 Auto Scaling 群組暫時移除執行個體](#)。

如果您選擇等待，則必須採取其他步驟以讓這些執行個體回到正常運作狀態。如果不這樣做，執行個體重新整理會取代所有 InService 執行個體，並等待一小時。然後，如果任何 Standby 執行個體仍然存在，執行個體重新整理就會失敗。為了避免此情況，請改為選擇忽略或終止這些執行個體。

- e. 對於受縮減保護的執行個體，請選擇忽略、取代或等待。這會決定找到受縮減保護的執行個體時所發生的情況。如需詳細資訊，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。

如果您選擇等待，則必須採取其他步驟以移除這些執行個體的縮減保護。如果不這樣做，執行個體重新整理會取代所有未受保護的執行個體，並等待一小時。然後，如果任何受縮減保護的執行個體仍然存在，執行個體重新整理就會失敗。為了避免此情況，請改為選擇忽略或取代這些執行個體。

6. (選用) 針對 CloudWatch 警示，選擇啟用 CloudWatch 警示，然後選擇一個或多個警示。CloudWatch 警示可用於識別任何問題，並在警示進入 ALARM 狀態時使作業失敗。如需詳細資訊，請參閱[使用自動復原啟動執行個體重新整理](#)。
7. (選用) 展開所需組態區段，指定您想要對 Auto Scaling 群組進行的更新。

對於此步驟，您可以選擇使用 JSON 或 YAML 語法來編輯參數值，而不用在主控台界面中進行選擇。若要執行此作業，請選擇 Use code editor (使用程式碼編輯器)，不要選擇 Use console interface (使用主控台界面)。下列程序說明如何使用主控台界面進行選擇。

- a. 針對 Update launch template (更新啟動範本)：

- 如果您尚未為 Auto Scaling 群組建立新啟動範本或新啟動範本版本，請勿勾選此核取方塊。
- 如果您建立了新啟動範本或新啟動範本版本，請勾選此核取方塊。當您選取此選項時，Amazon EC2 Auto Scaling 會顯示目前的啟動範本和目前的啟動範本版本。還會列出任何其他可用版本。依序選擇啟動範本和版本。

選擇版本後，便會看到版本資訊。這是執行個體重新整理過程中，取代執行個體時會使用的啟動範本版本。如果執行個體重新整理成功，則在新執行個體啟動時 (例如群組擴展時) 也會使用此版本的啟動範本。

- b. 針對 Choose a set of instance types and purchase options to override the instance type in the launch template (選擇一組執行個體類型和購買選項，以覆寫啟動範本中的執行個體類型)：

- 如果您希望使用您在啟動範本中指定的執行個體類型和購買選項，請勿勾選此核取方塊。
- 如果您希望複寫啟動範本中的執行個體類型或執行 Spot 執行個體，請勾選此核取方塊。您可以手動新增每個執行個體類型，或選擇一個主要執行個體類型和建議選項，用於為您擷取任何其他相符的執行個體類型。如果您打算啟動 Spot 執行個體，建議您新增幾種不同的執行個體類型。這樣，如果所選可用區域中的執行個體容量不足，則 Amazon EC2 Auto Scaling 可以啟動其他執行個體類型。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

 Warning

請勿將 Spot 執行個體與無法處理 Spot 執行個體中斷的應用程式搭配使用。如果 Amazon EC2 Spot 服務需要回收容量，則可能會發生中斷。

如果您選中此核取方塊，請確保啟動範本尚未請求 Spot 執行個體。您無法使用執行下列操作的啟動範本：請求 Spot 執行個體建立使用多個執行個體類型的 Auto Scaling 群組，以及啟動 Spot 執行個體與隨需執行個體。

 Note

如果要在目前使用啟動組態的 Auto Scaling 群組中設定這些選項，您必須在 Update launch template (更新啟動範本) 中選擇啟動範本。不支援在啟動組態中覆寫執行個體類型。

8. (選用) 對於復原設定，請選擇啟用自動復原，以在執行個體重新整理失敗時自動復原。

僅當 Auto Scaling 群組符合使用復原的先決條件時，才可以啟用此設定。

如需詳細資訊，請參閱[使用手動或自動轉返復原變更](#)。

9. 檢查您的所有選擇，以確認所有項目都已正確設定。

此時，最好驗證目前更改和提議更改之間的差異，確認差異是否會以意外或不想要的方式影響您的應用程式。若要確認您的執行個體類型是否與啟動範本相容，請參閱[執行個體類型相容性](#)。

10. 若您對執行個體重新整理的選取項目感到滿意，請選擇啟動執行個體重新整理。

在主控台中啟動執行個體重新整理 (混合執行個體群組)

如果您使用[混合執行個體政策](#)建立了 Auto Scaling 群組，請使用下列程序。如果您尚未為您的群組定義混合執行個體政策，請參閱[在主控台中啟動執行個體重新整理 \(基本程序\)](#)以啟動執行個體重新整理。

啟動執行個體重新整理

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Instance refresh (執行個體重新整理) 索引標籤的 Instance refreshes (執行個體重新整理) 中，選擇 Start instance refresh (啟動執行個體重新整理)。
4. 針對可用性設定，執行以下操作：
 - a. 針對執行個體取代方法：
 - 如果您尚未在 Auto Scaling 群組上設定執行個體維護政策，則執行個體取代方法的預設設定為終止並啟動。這是執行個體重新整理的舊版預設行為。
 - 如果您在 Auto Scaling 群組上設定了執行個體維護政策，則該政策會提供執行個體取代方法的預設值。若要覆寫執行個體維護政策，請選擇覆寫。覆寫功能僅適用於目前的執行個體重新整理。下次啟動執行個體重新整理時，這些值會重設為執行個體維護政策的預設值。

下列程序說明如何更新執行個體取代方法。

- i. 選擇下列其中一種執行個體取代方法：
 - 終止前啟動：必須先佈建新執行個體，才能終止現有執行個體。對於重視可用性多於節省成本的應用程式來說，這是理想的選擇。
 - 終止並啟動：在終止現有執行個體的同時佈建新執行個體。對於重視節省成本多於可用性的應用程式來說，這是理想的選擇。對於不應啟動超過目前可用容量的應用程式而言，這也是一個不錯的選擇。
 - 自訂行為：此選項可讓您根據想要在取代執行個體時的可用容量來設定自訂上下限範圍。這可協助您在成本和可用性之間取得適當的平衡。
- ii. 針對設定正常運作狀態百分比，請輸入下列一個或兩個欄位的值。啟用欄位會因您為執行個體取代方法所選的選項而異。

- 下限：設定繼續執行個體重整理所需的狀態良好最低百分比。
 - 上限：設定執行個體重整理時可能的狀態良好最高百分比。
- iii. 展開根據目前群組大小在取代期間檢視預估的臨時容量區段，以確認如何在群組中套用下限和上限。使用的實際值取決於所需的容量值，此值會隨群組擴展而有所改變。
 - iv. 展開設定無效取代大小的回復行為區段，然後選擇是否違反正常運作狀態百分比上限，以排列可用性的優先順序，或選擇違反正常運作狀態百分比下限。

對於非常小的群組，不建議保留預設的違反正常運作狀態百分比下限選項。如果 Auto Scaling 群組中只有一個執行個體，啟動執行個體重整理可能會導致中斷。

如果您使用的 Auto Scaling 群組尚未設定執行個體維護政策，則此步驟會設定回復行為。當您的群組有執行個體維護政策時，此選項無法使用，也不會顯示。另外，此選項僅適用於終止並啟動取代方法。其他取代方法將違反狀態良好最高百分比，以便優先考慮可用性。

- b. 在執行個體暖機期中，輸入從新執行個體的狀態變更為 InService 到完成初始化所經過的秒數。Amazon EC2 Auto Scaling 會等待這段時間，然後再繼續取代下一個執行個體。

暖機時，新啟動的執行個體也不會計入 Auto Scaling 群組的彙總執行個體指標 (例如 CPUUtilization、NetworkIn 和 NetworkOut)。如果將擴展政策新增至 Auto Scaling 群組，則擴展活動會平行執行。如果您為執行個體重整理暖機期設定了長間隔，則新啟動的執行個體在指標中顯示需要更多時間。因此，足夠的暖機期可防止 Amazon EC2 Auto Scaling 擴展過時的指標資料。

如果您已經正確定義 Auto Scaling 群組的預設執行個體暖機期，就不需要變更執行個體暖機期。但是，如果您想要覆寫預設值，您可以為此選項設定一個值。如需有關設定預設執行個體暖機期的詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

5. 針對重新整理設定，執行以下操作：

- a. (選用) 在 Checkpoints (檢查點) 欄位中，選擇 Enable checkpoints (啟用檢查點)，以使用對執行個體重整理的增量或分階段方法來取代執行個體。這會為每次取代之間的驗證提供額外時間。如果您選擇不啟用檢查點，則執行個體會在一次近乎連續的操作中被取代。

如果您要啟用檢查點，請參閱[啟用檢查點 \(主控台\)](#) 了解額外的步驟。

- b. 啟用或關閉 Skip matching (略過相符項目)：

- 若要略過取代已與啟動範本相符的執行個體以及任何執行個體類型覆寫，請讓啟用略過相符項目核取方塊保持為已勾選的狀態。

- 如果您選擇清除此核取方塊來關閉略過相符項目，則可取代所有執行個體。

當您啟用跳過比對時，您可以設定新啟動範本或啟動範本的新版本，而不是使用現有啟動範本。可以在啟動執行個體重新整理頁面的所需組態區段中執行這項操作。您也可以更新 Desired configuration (所需組態) 中更新執行個體類型覆寫。

- c. 對於待命執行個體，請選擇忽略、終止或等待。這會決定找到 Standby 狀態的執行個體時所發生的情況。如需詳細資訊，請參閱[從 Auto Scaling 群組暫時移除執行個體](#)。

如果您選擇等待，則必須採取其他步驟以讓這些執行個體回到正常運作狀態。如果不這樣做，執行個體重新整理會取代所有 InService 執行個體，並等待一小時。然後，如果任何 Standby 執行個體仍然存在，執行個體重新整理就會失敗。為了避免此情況，請改為選擇忽略或終止這些執行個體。

- d. 對於受縮減保護的執行個體，請選擇忽略、取代或等待。這會決定找到受縮減保護的執行個體時所發生的情況。如需詳細資訊，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。

如果您選擇等待，則必須採取其他步驟以移除這些執行個體的縮減保護。如果不這樣做，執行個體重新整理會取代所有未受保護的執行個體，並等待一小時。然後，如果任何受縮減保護的執行個體仍然存在，執行個體重新整理就會失敗。為了避免此情況，請改為選擇忽略或取代這些執行個體。

6. (選用) 針對 CloudWatch 警示，選擇啟用 CloudWatch 警示，然後選擇一個或多個警示。CloudWatch 警示可用於識別任何問題，並在警示進入 ALARM 狀態時使作業失敗。如需詳細資訊，請參閱[使用自動復原啟動執行個體重新整理](#)。
7. 在 Desired configuration (所需組態) 區段中，執行下列操作。

對於此步驟，您可以選擇使用 JSON 或 YAML 語法來編輯參數值，而不用在主控台界面中進行選擇。若要執行此作業，請選擇 Use code editor (使用程式碼編輯器)，不要選擇 Use console interface (使用主控台界面)。下列程序說明如何使用主控台界面進行選擇。

- a. 針對 Update launch template (更新啟動範本)：

- 如果您尚未為 Auto Scaling 群組建立新啟動範本或新啟動範本版本，請勿勾選此核取方塊。
- 如果您建立了新啟動範本或新啟動範本版本，請勾選此核取方塊。當您選取此選項時，Amazon EC2 Auto Scaling 會顯示目前的啟動範本和目前的啟動範本版本。還會列出任何其他可用版本。依序選擇啟動範本和版本。

選擇版本後，便會看到版本資訊。這是執行個體重新整理過程中，取代執行個體時會使用的啟動範本版本。如果執行個體重新整理成功，則在新執行個體啟動時 (例如群組擴展時) 也會使用此版本的啟動範本。

- b. 針對 Use these settings to override the instance type and purchase option defined in the launch template (使用這些設定覆寫啟動範本中所定義的執行個體類型和購買選項)：

根據預設，會選取此核取方塊。Amazon EC2 Auto Scaling 會使用目前在 Auto Scaling 群組的混合執行個體政策中設定的值填入每個參數。僅針對您想要變更的參數來更新參數值。如需這些設定的指導，請參閱 [具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

Warning

建議您不要清除此核取方塊。只有當您想停止使用混合執行個體政策時才清除它。在執行個體重新整理成功後，Amazon EC2 Auto Scaling 會更新您的群組，以符合 Desired configuration (所需組態)。如果 Amazon EC2 Auto Scaling 不再包含混合執行個體政策，則會逐步終止目前正在執行的所有 Spot 執行個體，並將其取代為隨需執行個體。或者，如果您的啟動範本請求 Spot 執行個體，則 Amazon EC2 Auto Scaling 會逐漸終止目前執行的所有隨需執行個體，並用 Spot 執行個體取代它們。

8. (選用) 對於復原設定，請選擇啟用自動復原，以在執行個體重新整理因任何原因失敗時自動復原。

僅當 Auto Scaling 群組符合使用復原的先決條件時，才可以啟用此設定。

如需詳細資訊，請參閱[使用手動或自動轉返復原變更](#)。

9. 檢查您的所有選擇，以確認所有項目都已正確設定。

此時，最好驗證目前更改和提議更改之間的差異，確認差異是否會以意外或不想要的方式影響您的應用程式。若要確認您的執行個體類型是否與啟動範本相容，請參閱[執行個體類型相容性](#)。

若您對執行個體重新整理的選取項目感到滿意，請選擇啟動執行個體重新整理。

啟動執行個體重新整理 (AWS CLI)

啟動執行個體重新整理

請使用 [start-instance-refresh](#) 命令，從 AWS CLI 啟動執行個體重新整理。您可以指定要在 JSON 組態檔案中變更的任何偏好設定。當您參考組態檔案時，請提供檔案路徑和名稱，如下列範例所示。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容：

```
{
  "AutoScalingGroupName": "my-asg",
  "Preferences": {
    "InstanceWarmup": 60,
    "MinHealthyPercentage": 50,
    "AutoRollback": true,
    "ScaleInProtectedInstances": Ignore,
    "StandbyInstances": Terminate
  }
}
```

如果未提供偏好設定，則會使用預設值。如需詳細資訊，請參閱[了解執行個體重新整理的預設值](#)。

輸出範例：

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

使用 AWS 管理主控台 或 監控執行個體重新整理 AWS CLI

您可以使用 AWS 管理主控台 或 監控進行中執行個體重新整理，或查詢過去六週內執行個體重新整理的狀態 AWS CLI。

監控並檢查執行個體重新整理的狀態

若要監控和檢查執行個體重新整理的狀態，請使用下列其中一種方法：

Console

Tip

在此程序中，應該已顯示具名資料欄。若要顯示隱藏的資料欄或變更顯示的資料列數，請選擇區段右上角的齒輪圖示以開啟偏好設定模式。視需要更新設定，然後選擇確認。

監控和檢查執行個體重新整理的狀態（主控台）

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Instance refresh (執行個體重新整理) 索引標籤的 Instance refreshes history (執行個體重新整理歷史記錄) 下，您可以查看 Status (狀態) 資料欄來判斷請求的狀態。操作在初始化時會進入 Pending 狀態。然後狀態應該會快速變更為 InProgress。當所有執行個體都更新後，狀態會變更為 Successful。
4. 您可以檢視群組的擴展活動，進一步監控進行中活動的成功或失敗。在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 下，當執行個體重新整理開始時，您會在執行個體終止時看到項目，並在執行個體啟動時看到另一組項目。如果您有許多擴展活動，您可以選擇活動歷史記錄頂端的 > 圖示來查看更多。如需有關對可能導致活動失敗的問題進行故障診斷的資訊，請參閱 [對 Amazon EC2 Auto Scaling 中的問題進行故障診斷](#)。
5. （選用）在執行個體管理索引標籤的執行個體下，您可以視需要檢閱特定執行個體的進度。

AWS CLI

監控和檢查執行個體重新整理的狀態 (AWS CLI)

使用下列 [describe-instance-refreshes](#) 命令。

```
aws autoscaling describe-instance-refreshes --auto-scaling-group-name my-asg
```

以下為範例輸出。

執行個體重新整理會依開始時間排序。首先說明仍在進行中的執行個體重新整理。

```
{
  "InstanceRefreshes": [
    {
      "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b",
      "AutoScalingGroupName": "my-asg",
      "Status": "InProgress",
      "StatusReason": "Waiting for instances to warm up before continuing. For example: i-0645704820a8e83ff is warming up.",
    }
  ]
}
```

```

    "StartTime": "2023-11-24T16:46:52+00:00",
    "PercentageComplete": 50,
    "InstancesToUpdate": 0,
    "Preferences": {
      "MaxHealthyPercentage": 120,
      "MinHealthyPercentage": 90,
      "InstanceWarmup": 60,
      "SkipMatching": false,
      "AutoRollback": true,
      "ScaleInProtectedInstances": "Ignore",
      "StandbyInstances": "Ignore"
    }
  },
  {
    "InstanceRefreshId": "0e151305-1e57-4a32-a256-1fd14157c5ec",
    "AutoScalingGroupName": "my-asg",
    "Status": "Successful",
    "StartTime": "2023-11-22T13:53:37+00:00",
    "EndTime": "2023-11-22T13:59:45+00:00",
    "PercentageComplete": 100,
    "InstancesToUpdate": 0,
    "Preferences": {
      "MaxHealthyPercentage": 120,
      "MinHealthyPercentage": 90,
      "InstanceWarmup": 60,
      "SkipMatching": false,
      "AutoRollback": true,
      "ScaleInProtectedInstances": "Ignore",
      "StandbyInstances": "Ignore"
    }
  }
]
}

```

您可以檢視群組的擴展活動，進一步監控進行中活動的成功或失敗。擴展活動也可協助您深入了解更多詳細資訊，以協助您針對執行個體重新整理的問題進行疑難排解。如需詳細資訊，請參閱[對 Amazon EC2 Auto Scaling 中的問題進行故障診斷](#)。

執行個體重新整理狀態

當您啟動執行個體重新整理時，它會進入擱置中狀態。它會從擱置中傳遞至進行中，直到達到成功、失敗、已取消、復原成功或復原失敗為止。

執行個體重新整理可以具有下列狀態：

狀態	Description
待定	已建立請求，但執行個體重新整理尚未啟動。
InProgress	執行個體重新整理正在進行中。
成功	執行個體重新整理已成功完成。
失敗	無法完成執行個體重新整理。您可以使用狀態原因和擴展活動進行疑難排解。
取消	正在取消進行中的執行個體重新整理。
已取消	執行個體重新整理已取消。
復原進行中	正在復原執行個體重新整理。
復原失敗	復原無法完成。您可以使用狀態原因和擴展活動進行疑難排解。
復原成功	復原已成功完成。
烘焙	執行個體重新整理完成更新執行個體後，等待指定的製作時間。

在執行個體重新整理期間取代根磁碟區

目錄

- [運作方式](#)
- [要求](#)
- [使用取代根磁碟區啟動執行個體重新整理](#)
- [限制](#)
- [使用生命週期掛鉤取代根磁碟區](#)

運作方式

根磁碟區取代只會取代根 EBS 磁碟區，同時讓執行個體保持執行狀態，以更新您的執行個體。這消除了啟動新執行個體的需求，並避免潛在的容量限制。此程序會保留所有其他執行個體資源，包括：

- 網路介面和 IP 地址
- 非根 EBS 磁碟區
- 執行個體存放區磁碟區和資料
- 安全群組和 IAM 角色

在根磁碟區取代期間，您的應用程式會繼續在現有執行個體上執行。原始根磁碟區會分離，從您指定的 AMI 建立新的根磁碟區，然後連接至相同的執行個體。原始根磁碟區會在成功取代後自動刪除。

當您使用取代根磁碟區策略時，執行個體會經歷以下生命週期狀態：

1. `ReplacingRootVolume` - 根磁碟區取代開始
2. `ReplacingRootVolume:Wait` - 等待生命週期掛鉤完成（如果已設定）
3. `ReplacingRootVolume:Proceed` - 繼續替換
4. `RootVolumeReplaced` - 已成功完成取代

根磁碟區取代完成後，Auto Scaling 會對更新的執行個體執行運作狀態檢查。如果執行個體的根磁碟區取代失敗，Auto Scaling 會將該執行個體標記為終止，並以新的執行個體取代。

要求

- Auto Scaling 群組必須使用混合執行個體政策
- 混合執行個體政策中的所有覆寫都必須指定 `ImageId`
- AMIs 只能包含單一根磁碟區
- 所有執行個體都必須符合群組的啟動範本組態
- 您必須使用具有包含 `ImageId` 覆寫之混合執行個體政策的所需組態來啟動執行個體重新整理。

使用取代根磁碟區啟動執行個體重新整理

使用取代根磁碟區 (AWS CLI) 啟動執行個體重新整理

使用下列 [start-instance-refresh](#) 命令，從 啟動執行個體重新整理 AWS CLI。您可以指定要在 JSON 組態檔案中變更的任何偏好設定。當您參考組態檔案時，請提供檔案路徑和名稱，如下列範例所示。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容：

Example

```
{
  "AutoScalingGroupName": "my-asg",
  "Strategy" : "ReplaceRootVolume",
  "DesiredConfiguration":{
    "MixedInstancesPolicy":{
      "LaunchTemplate": {
        "LaunchTemplateSpecification": {
          "LaunchTemplateId":"my-launch-template",
          "Version": "1"
        },
        "Overrides": [
          {
            "InstanceType":"c5.large",
            "ImageId":"ami-1234example"
          },
          {
            "InstanceType":"m5.large",
            "ImageId":"ami-2345example"
          }
        ]
      }
    }
  },
  "Preferences": {
    "InstanceWarmup": 60,
    "MinHealthyPercentage": 90,
    "AutoRollback": true,
    "ScaleInProtectedInstances": Ignore,
    "StandbyInstances": Ignore
  }
}
```

如果未提供偏好設定，則會使用預設值。如需詳細資訊，請參閱[了解執行個體重新整理的預設值](#)。

輸出範例：

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
```



```
}
```

限制

- 如果 EC2 Auto Scaling 群組或執行個體重新整理所需的組態使用 `$Latest` 或 `$Default` 啟動範本版本，則無法使用取代根磁碟區啟動執行個體重新整理。
- 如果執行個體包含執行個體類型不存在於混合執行個體政策中的執行個體，則無法使用取代 Amazon EC2 Auto Scaling 群組上的根磁碟區來啟動執行個體重新整理。
- 使用取代根磁碟區重新整理執行個體期間，您只能連接具有所需組態混合執行個體政策中存在之類型的執行個體。
- 您無法在具有暖集區的 EC2 Auto Scaling 群組上使用取代根磁碟區來啟動執行個體重新整理。
- 您無法將暖集區新增至使用取代根磁碟區重新整理作用中執行個體的 Amazon EC2 Auto Scaling 群組。

使用生命週期掛鉤取代根磁碟區

正在進行根磁碟區取代的執行個體會遵循自己的生命週期轉換，可讓您在取代之前和/或之後叫用動作（例如 Lambda 函數）。如需根磁碟區取代期間生命週期狀態的詳細資訊，請參閱 [正在進行根磁碟區取代之執行個體的生命週期狀態轉換](#)。

如需有關新增 lifecycle hook 的詳細資訊，請參閱 [將生命週期掛鉤新增至 Auto Scaling 群組](#)。如需有關完成生命週期動作的詳細資訊，請參閱 [在 Auto Scaling 群組中完成生命週期動作](#)。

在根磁碟區取代之前，生命週期關聯在下列案例中非常有用：

- 您想要在取代根磁碟區並重新啟動執行個體之前，正常關閉應用程式。
- 您想要在取代磁碟區之前，將資料從根磁碟區移至另一個位置。

在根磁碟區取代之後，生命週期掛鉤在下列案例中非常有用：

- 您想要在新 AMI 開始接收流量之前，確認執行個體已完全就緒。
- 您想要引導先前存在於根磁碟區上的應用程式資料。

在新增 lifecycle hook 時，請考慮以下事項：

- 為生命週期動作設定 `autoscaling:EC2_INSTANCE_TERMINATING` 生命週期掛鉤時，即將進行根磁碟區取代的執行個體會達到 `ReplacingRootVolume:Wait` 狀態時暫停，以執行自訂動作。

- 為生命週期動作設定autoscaling:EC2_INSTANCE_LAUNCHING生命週期掛鉤時，剛取代其根磁碟區的執行個體會達到 Pending:Wait 狀態時暫停，以執行自訂動作。

當執行個體進入等待狀態時，Amazon EC2 Auto Scaling 會傳送通知。Auto Scaling 生命週期掛鉤通知包含值為 Action的欄位ReplaceRootVolume，表示執行個體正在進行根磁碟區取代。

有關這些通知的範例，請參閱本指南的「EventBridge」章節。如需更多資訊，請參閱[執行個體重新整理取代根磁碟區生命週期事件](#)。

使用 AWS 管理主控台 或 取消執行個體重新整理 AWS CLI

您可以取消仍在進行中的執行個體重新整理。該作業完成後即無法取消。

取消執行個體重新整理並不會復原任何已取代的執行個體。若要復原對執行個體所進行的變更，請改為執行復原。如需詳細資訊，請參閱[使用手動或自動轉返復原變更](#)。

主題

- [取消執行個體重新整理 \(主控台\)](#)
- [取消執行個體重新整理 \(AWS CLI\)](#)

取消執行個體重新整理 (主控台)

取消執行個體重新整理

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。
3. 在執行個體重新整理索引標籤的作用中執行個體重新整理中，選擇動作 > 取消。
4. 出現確認提示時，選擇 Confirm (確認)。

執行個體重新整理的狀態隨即設定為取消中。取消完成後，執行個體重新整理的狀態就會設定為已取消。

取消執行個體重新整理 (AWS CLI)

取消執行個體重新整理

使用 中的 [cancel-instance-refresh](#) 命令 AWS CLI，並提供 Auto Scaling 群組名稱。

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg
```

輸出範例：

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

若要取消執行個體重新整理，而不等待轉換執行個體

使用 `--no-wait-for-transitioning-instances` 選項搭配來自的 [cancel-instance-refresh](#) 命令 AWS CLI，取消執行個體重新整理，而無需等待轉換執行個體。

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg --no-wait-for-transitioning-instances
```

根據預設，當您取消執行個體重新整理時，Amazon EC2 Auto Scaling 會等待任何進行中的啟動和終止完成，然後再將狀態設定為已取消。如果您有長時間耗盡的執行個體，或需要立即啟動新的執行個體重新整理，您可以使用 `命令`，讓您不必等待這些轉換操作。

當您使用此選項時，執行個體重新整理狀態會變更為已取消，讓您啟動新的執行個體重新整理。任何進行中的啟動和終止都會在背景繼續。

使用手動或自動轉返復原變更

您可以復原仍在進行中的執行個體重新整理。該作業完成後即無法復原。不過，您可以透過啟動新的執行個體重新整理，再次更新 Auto Scaling 群組。

復原時，Amazon EC2 Auto Scaling 會取代到目前為止已部署的執行個體。新執行個體會與您在啟動執行個體重新整理之前，上次儲存在 Auto Scaling 群組中的組態相符。

Amazon EC2 Auto Scaling 提供下列復原方式：

- 手動復原：您可以手動啟動復原，以撤銷已部署到復原點的內容。
- 自動復原：如果執行個體重新整理因某些原因而失敗，或您指定的任何 CloudWatch 警示進入 ALARM 狀態，Amazon EC2 Auto Scaling 會自動撤銷已部署的內容。

目錄

- [考量事項](#)
- [手動啟動復原](#)
- [使用自動復原啟動執行個體重新整理](#)

考量事項

使用復原時需要注意以下事項：

- 只有在您指定了所需的組態作為啟動執行處理重新整理的一部分時，才能使用復原選項。
- 如果啟動範本的版本為特定編號版本，您只能復原至一個先前版本。如果將 Auto Scaling 群組設定為使用 `$Latest` 或 `$Default` 啟動範本版本，則無法使用復原選項。
- 您也無法轉返至設定為使用 AWS Systems Manager 參數存放區的 AMI 別名的啟動範本。
- 您上次儲存在 Auto Scaling 群組中的組態必須處於穩定狀態。如果它未處於穩定狀態，仍會進行復原工作流程，但會以失敗告終。在您解決此問題之前，Auto Scaling 群組可能處於失敗狀態，無法再成功啟動執行個體。這可能會影響服務或應用程式的可用性。

手動啟動復原

Console

手動啟動執行個體重新整理的復原 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。
3. 在執行個體重新整理索引標籤的作用中執行個體重新整理中，選擇動作 > 啟動復原。
4. 出現確認提示時，選擇 Confirm (確認)。

AWS CLI

手動啟動執行個體重新整理的復原 (AWS CLI)

使用 AWS CLI 中的 [rollback-instance-refresh](#) 命令，並提供 Auto Scaling 群組名稱。

```
aws autoscaling rollback-instance-refresh --auto-scaling-group-name my-asg
```

輸出範例：

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

 Tip

如果此命令擲回錯誤，請確定您已在 AWS CLI 本機將 更新為最新版本。

使用自動復原啟動執行個體重新整理

使用自動復原功能，您便可以在執行個體重新整理失敗時 (例如發生錯誤或指定的 Amazon CloudWatch 警示進入 ALARM 狀態時) 自動復原執行個體重新整理。

如果您啟用自動復原功能，且在取代執行個體時發生錯誤，則執行個體重新整理會在失敗前一小時內嘗試完成所有取代作業，然後復原。這些錯誤通常是由 EC2 啟動失敗、錯誤設定的運作狀態檢查，或者未略過或允許處於 Standby 狀態或受縮減保護的執行個體終止等原因所引起。

您可以選擇是否要指定 CloudWatch 警示。若要指定警示，您需要先建立警示。您可以指定指標警示和複合警示。如需建立警示的詳細資訊，請參閱《[Amazon CloudWatch 使用者指南](#)》。以 Elastic Load Balancing 指標為例，如果您使用 Application Load Balancer，便可以使用 HTTPCode_ELB_5XX_Count 和 HTTPCode_ELB_4XX_Count 指標。

考量事項

- 如果您指定 CloudWatch 警示，但未啟用自動復原，且警示狀態進入 ALARM，則執行個體重新整理會失敗而不會復原。
- 啟動執行個體重新整理時，您最多可以選擇 10 個警示。
- 選擇 CloudWatch 警示時，警示必須處於相容狀態。如果警示狀態為 INSUFFICIENT_DATA 或 ALARM，您會在嘗試啟動執行個體重新整理時收到錯誤訊息。
- 建立要使用的 Amazon EC2 Auto Scaling 警示時，警示應包括如何處理遺失的資料點。如果指標本來就會經常遺失資料點，則警示的狀態在這些期間為 INSUFFICIENT_DATA。發生這種情況時，Amazon EC2 Auto Scaling 無法取代執行個體，直到找到新的資料點為止。為了強制警示維持先前的 ALARM 或 OK 狀態，您可以選擇忽略遺失的資料。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[設定警示如何處理遺失資料](#)。

Console

使用自動復原啟動執行個體重新整理 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。
3. 在 Instance refresh (執行個體重新整理) 索引標籤的 Instance refreshes (執行個體重新整理) 中，選擇 Start instance refresh (啟動執行個體重新整理)。
4. 遵循 [啟動執行個體重新整理 \(主控台\)](#) 的程序操作，並視需要設定執行個體重新整理。
5. (選用) 在重新整理設定下，針對 CloudWatch 警示，選擇啟用 CloudWatch 警示，然後選擇一個或多個警示以識別任何問題，並在警示進入 ALARM 狀態時使作業失敗。
6. 在復原設定下，選擇啟用自動復原，以便在開始執行個體重新整理之前，自動將失敗的執行個體重新整理復原至您上次儲存在 Auto Scaling 群組的設定。
7. 檢視您的選擇，然後選擇啟動執行個體重新整理。

AWS CLI

使用自動復原啟動執行個體重新整理 (AWS CLI)

使用 [start-instance-refresh](#) 命令，並為 Preferences 中的 AutoRollback 選項指定 true。

下列範例示範如何在啟動執行個體重新整理時一旦發生失敗就會自動復原。以自訂值取代 *italicized* 參數值。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容。

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1"
    }
  },
  "Preferences": {
```

```
    "AutoRollback": true
  }
}
```

或者，若要在執行個體重新整理失敗或指定的 CloudWatch 警示處於 ALARM 狀態時自動復原，請在 Preferences 中指定 AlarmSpecification 選項並提供警示名稱，如以下範例所示。以自訂值取代 *italicized* 參數值。

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1"
    }
  },
  "Preferences": {
    "AutoRollback": true,
    "AlarmSpecification": { "Alarms": [ "my-alarm" ] }
  }
}
```

如果成功，此命令傳回的輸出會類似如下。

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

Tip

如果此命令擲回錯誤，請確定您已在 AWS CLI 本機將 更新為最新版本。

搭配略過相符項目使用執行個體重新整理

略過相符項目會讓 Amazon EC2 Auto Scaling 忽略已經有最新更新的執行個體。如此一來，您就不會取代超過需要的執行個體數量。當您想要確定 Auto Scaling 群組使用特定版本的啟動範本，並且僅取代使用不同版本的執行個體時，這會很有幫助。

略過相符項目有下列考量：

- 如果同時使用略過相符項目和所需組態來啟動執行個體重新整理，Amazon EC2 Auto Scaling 會檢查是否有任何執行個體與所需組態相符。然後，它只會取代與您所需組態不相符的執行個體。執行個體重新整理成功後，Amazon EC2 Auto Scaling 會更新群組，以反映所需組態。
- 如果您使用略過相符項目來啟動執行個體重新整理，但未指定所需組態，Amazon EC2 Auto Scaling 會檢查是否有任何執行個體與您上次儲存在 Auto Scaling 群組上的組態相符。然後，它只會取代與您上次儲存的組態不相符的執行個體。
- 您可以將略過相符項目與新啟動範本、啟動範本新版本或一組執行個體類型搭配使用。如果啟用略過相符項目功能，但這些都沒有變更，則執行個體重新整理會立即成功，且不會取代任何執行個體。如果您對所需組態進行了任何其他變更 (例如變更 Spot 分配策略)，Amazon EC2 Auto Scaling 會等待執行個體重新整理成功完成。然後，它會更新 Auto Scaling 群組設定，以反映新的所需組態。
- 您無法將略過相符項目功能與新的啟動組態搭配使用。
- 當您啟動執行個體重新整理並提供所需的組態時，Amazon EC2 Auto Scaling 會確保所有執行個體都使用您所需的組態。因此，當您指定 `$Default` 或 `$Latest` 做為啟動範本的所需版本，然後在執行個體重新整理進行時建立新的啟動範本版本時，任何已取代的執行個體都會再次取代。
- 略過比對並不知道啟動範本中的使用者資料指令碼是否會提取更新的程式碼，並將其安裝在新的執行個體上。因此，略過比對可能會略過取代已安裝過時程式碼的執行個體。在此情況下，您應該關閉略過相符項目，以確保即使沒有啟動範本版本更新，所有執行個體都會收到最新的程式碼。

本節包含啟動已啟用略過比對的執行個體重新整理 AWS CLI 的指示。如需使用主控台的說明，請參閱「[啟動執行個體重新整理 \(主控台\)](#)」。

略過相符項目 (基本程序)

請依照本節中的步驟，使用 AWS CLI 執行下列動作：

- 建立您要套用到執行個體的啟動範本。
- 啟動執行個體重新整理，以將啟動範本套用到 Auto Scaling 群組。如果您沒有啟用略過相符項目，則會取代所有執行個體。即使用於佈建執行個體的啟動範本與您為所需組態指定的啟動範本相同，也是如此。

將略過相符項目功能與新啟動範本搭配使用

1. 使用 [create-launch-template](#) 命令來為 Auto Scaling 群組建立新的啟動範本。請包含 `--launch-template-data` 選項，以及定義已為您 Auto Scaling 群組建立的執行個體詳細資訊的 JSON 輸入。

例如，使用下列命令建立具有 AMI ID `ami-0123456789abcdef0` 和 `t2.micro` 執行個體類型的基本啟動範本。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
--version-description version1 \
--launch-template-data
'{"ImageId": "ami-0123456789abcdef0", "InstanceType": "t2.micro"}
```

如果成功，此命令傳回的輸出會類似如下。

```
{
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-068f72b729example",
    "LaunchTemplateName": "my-template-for-auto-scaling",
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "CreateTime": "2023-01-30T18:16:06.000Z",
    "DefaultVersionNumber": 1,
    "LatestVersionNumber": 1
  }
}
```

如需詳細資訊，請參閱[使用 建立和管理啟動範本的範例 AWS CLI](#)。

2. 使用 [start-instance-refresh](#) 命令來啟動執行個體取代工作流程，並套用具有 ID `lt-068f72b729example` 的新啟動範本。由於啟動範本是全新內容，因此只有一個版本。這表示啟動範本第 1 版是此執行個體重新整理的目標。如果在執行個體重新整理期間發生橫向擴展事件，而 Amazon EC2 Auto Scaling 使用此第 1 版啟動範本佈建新執行個體，它們則不會遭到取代。成功完成此操作後，新啟動範本便已成功套用到 Auto Scaling 群組。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容。

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "LaunchTemplate": {
      "LaunchTemplateId": "lt-068f72b729example",
      "Version": "$Default"
    }
  }
}
```

```
    },
    "Preferences": {
      "SkipMatching": true
    }
  }
```

如果成功，此命令傳回的輸出會類似如下。

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

略過相符項目 (混合執行個體群組)

如果您有具有[混合執行個體政策](#)的 Auto Scaling 群組，請依照本節中的步驟使用 AWS CLI 來啟動執行個體重新整理，並略過相符項目。您有下列選項：

- 提供新啟動範本，以套用到政策中指定的所有執行個體類型。
- 提供一組更新的執行個體類型 (不論是否變更政策中的啟動範本)。例如，您可能會想要從不需要的執行個體類型遷移。您可以依現狀使用啟動範本，而不變更要取代之執行個體的 AMI、安全群組或其他具體內容。

根據符合您需求的選項，依照下列其中一節中的步驟進行操作。

將略過相符項目功能與新啟動範本搭配使用

1. 使用 [create-launch-template](#) 命令來為 Auto Scaling 群組建立新的啟動範本。請包含 `--launch-template-data` 選項，以及定義已為您 Auto Scaling 群組建立的執行個體詳細資訊的 JSON 輸入。

例如，使用下列命令建立具有 AMI ID `ami-0123456789abcdef0` 的啟動範本。

```
aws ec2 create-launch-template --launch-template-name my-new-template --version-
description version1 \
  --launch-template-data '{"ImageId":"ami-0123456789abcdef0"}'
```

如果成功，此命令傳回的輸出會類似如下。

```
{
```

```
"LaunchTemplate": {
  "LaunchTemplateId": "lt-04d5cc9b88example",
  "LaunchTemplateName": "my-new-template",
  "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
  "CreateTime": "2023-01-31T15:56:02.000Z",
  "DefaultVersionNumber": 1,
  "LatestVersionNumber": 1
}
```

如需詳細資訊，請參閱[使用 建立和管理啟動範本的範例 AWS CLI](#)。

- 若要檢視 Auto Scaling 群組的現有混合執行個體政策，請執行 [describe-auto-scaling-groups](#) 命令。當您啟動執行個體重新整理時，您在下一個步驟中會需要此資訊。

下列命令範例會傳回針對名為 *my-asg* 之 Auto Scaling 群組所設定的混合執行個體政策。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

如果成功，此命令傳回的輸出會類似如下。

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn",
      "MixedInstancesPolicy": {
        "LaunchTemplate": {
          "LaunchTemplateSpecification": {
            "LaunchTemplateId": "lt-073693ed27example",
            "LaunchTemplateName": "my-old-template",
            "Version": "$Default"
          },
          "Overrides": [
            {
              "InstanceType": "c5.large"
            },
            {
              "InstanceType": "c5a.large"
            },
            {
              "InstanceType": "m5.large"
            }
          ]
        }
      }
    }
  ]
}
```

```

        {
            "InstanceType": "m5a.large"
        }
    ],
    "InstancesDistribution": {
        "OnDemandAllocationStrategy": "prioritized",
        "OnDemandBaseCapacity": 1,
        "OnDemandPercentageAboveBaseCapacity": 50,
        "SpotAllocationStrategy": "price-capacity-optimized"
    }
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 4,
...
}
]
}

```

3. 使用 [start-instance-refresh](#) 命令來啟動執行個體取代工作流程，並套用具有 ID *lt-04d5cc9b88example* 的新啟動範本。由於啟動範本是全新內容，因此只有一個版本。這表示啟動範本第 1 版是此執行個體重新整理的目標。如果在執行個體重新整理期間發生橫向擴展事件，而 Amazon EC2 Auto Scaling 使用此第 1 版啟動範本佈建新執行個體，它們則不會遭到取代。成功完成此操作後，更新的混合執行個體政策便已成功套用到 Auto Scaling 群組。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容。

```

{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "MixedInstancesPolicy": {
      "LaunchTemplate": {
        "LaunchTemplateSpecification": {
          "LaunchTemplateId": "lt-04d5cc9b88example",
          "Version": "$Default"
        },
        "Overrides": [
          {
            "InstanceType": "c5.large"
          }
        ]
      }
    }
  }
}

```

```

    },
    {
      "InstanceType": "c5a.large"
    },
    {
      "InstanceType": "m5.large"
    },
    {
      "InstanceType": "m5a.large"
    }
  ]
},
"InstancesDistribution": {
  "OnDemandAllocationStrategy": "prioritized",
  "OnDemandBaseCapacity": 1,
  "OnDemandPercentageAboveBaseCapacity": 50,
  "SpotAllocationStrategy": "price-capacity-optimized"
}
}
},
"Preferences": {
  "SkipMatching": true
}
}

```

如果成功，此命令傳回的輸出會類似如下。

```

{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}

```

在接下來的這個程序中，您會提供一組更新的執行個體類型，而不變更啟動範本。

將略過相符項目與一組更新的執行個體類型搭配使用

- 若要檢視 Auto Scaling 群組的現有混合執行個體政策，請執行 [describe-auto-scaling-groups](#) 命令。當您啟動執行個體重新整理時，您在下一個步驟中會需要此資訊。

下列命令範例會傳回針對名為 *my-asg* 之 Auto Scaling 群組所設定的混合執行個體政策。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

如果成功，此命令傳回的輸出會類似如下。

```
{
  "AutoScalingGroups":[
    {
      "AutoScalingGroupName":"my-asg",
      "AutoScalingGroupARN":"arn",
      "MixedInstancesPolicy":{"
        "LaunchTemplate":{"
          "LaunchTemplateSpecification":{"
            "LaunchTemplateId":"lt-073693ed27example",
            "LaunchTemplateName":"my-template-for-auto-scaling",
            "Version":"$Default"
          },
          "Overrides":[
            {
              "InstanceType":"c5.large"
            },
            {
              "InstanceType":"c5a.large"
            },
            {
              "InstanceType":"m5.large"
            },
            {
              "InstanceType":"m5a.large"
            }
          ]
        },
        "InstancesDistribution":{"
          "OnDemandAllocationStrategy":"prioritized",
          "OnDemandBaseCapacity":1,
          "OnDemandPercentageAboveBaseCapacity":50,
          "SpotAllocationStrategy":"price-capacity-optimized"
        }
      },
      "MinSize":1,
      "MaxSize":5,
      "DesiredCapacity":4,
      ...
    }
  ]
}
```

$$\left. \begin{array}{l} \} \\] \\ \} \end{array} \right\}$$

2. 使用 [start-instance-refresh](#) 命令來啟動執行個體取代工作流程，並套用您的更新。若要取代使用特定執行個體類型的執行個體，所需組態必須指定僅包含您需要的執行個體類型的混合執行個體政策。您可以選擇是否在原處新增新的執行個體類型。

下列命令範例會啟動執行個體重新整理，而不包含不需要的執行個體類型 *m5a.large*。當群組中的某個執行個體類型與剩下三個執行個體類型的其中一個不相符時，執行個體就會遭到取代。(請注意，執行個體重新整理不會選擇要從中佈建新執行個體的執行個體類型，[分配策略](#)才會這樣做。) 成功完成此操作後，更新的混合執行個體政策便已成功套用到 Auto Scaling 群組。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "MixedInstancesPolicy": {
      "LaunchTemplate": {
        "LaunchTemplateSpecification": {
          "LaunchTemplateId": "lt-073693ed27example",
          "Version": "$Default"
        },
        "Overrides": [
          {
            "InstanceType": "c5.large"
          },
          {
            "InstanceType": "c5a.large"
          },
          {
            "InstanceType": "m5.large"
          }
        ]
      },
      "InstancesDistribution": {
        "OnDemandAllocationStrategy": "prioritized",
        "OnDemandBaseCapacity": 1,
        "OnDemandPercentageAboveBaseCapacity": 50,
```

```
        "SpotAllocationStrategy": "price-capacity-optimized"
      }
    }
  },
  "Preferences": {
    "SkipMatching": true
  }
}
```

將檢查點新增至執行個體重新整理

使用執行個體重新整理時，您可以選擇分階段取代執行個體，以便隨時對執行個體執行驗證。若要分階段執行取代，您可以新增檢查點，即執行個體重新整理暫停的時間點。使用檢查點可讓您更好地控制選擇更新 Auto Scaling 群組的方式。它可幫助您確認應用程式以可靠、可預測的方式運作。

目錄

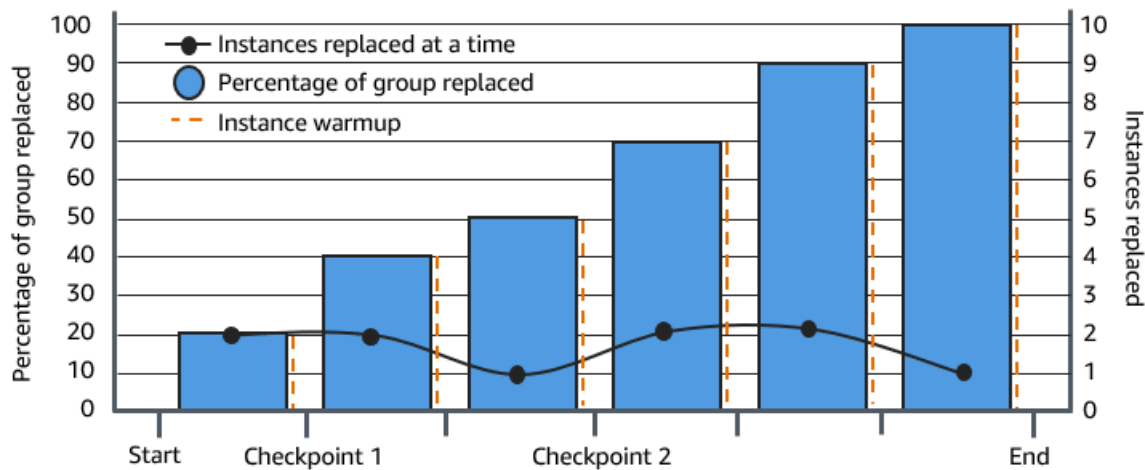
- [運作方式](#)
- [考量事項](#)
- [使用 AWS 管理主控台 或 啟用檢查點 AWS CLI](#)

運作方式

開始執行個體重新整理時，請以 Auto Scaling 群組中執行個體總數的百分比來指定檢查點。這些檢查點指出 Auto Scaling 群組中執行個體的最小百分比，在檢查點被視為達到之前必須是新的執行個體。例如，如果您的檢查點是 [20, 50, 100]，當 20% 的執行個體是新的、50% 是新的，以及所有執行個體都是新的最終檢查點時，第一個檢查點就會到達。

Amazon EC2 Auto Scaling 會逐步取代執行個體，以符合指定的檢查點百分比，同時維持群組的運作狀態百分比下限。為了達到檢查點百分比，Amazon EC2 Auto Scaling 有時會取代較少的執行個體，但絕不會超過狀態良好最低百分比允許的比例。

請考慮下列具有 10 個執行個體的 Auto Scaling 群組。檢查點百分比為 [20, 50, 100]，狀態良好最低百分比為 80%，狀態良好最高百分比為 100%。為了維持狀態良好最低百分比，一次只會取代兩個執行個體。下圖概括了到達檢查點之前取代執行個體的程序。



在上述範例中，每個啟動的新執行個體都有一個執行個體暖機期。您也可能有一個 lifecycle hook，會使執行個體進入等待狀態，然後在啟動或終止時執行自訂動作。

Amazon EC2 Auto Scaling 會為每個檢查點發出事件，但 100% 完成檢查點除外。您可以新增 EventBridge 規則，將事件傳送到 Amazon SNS 等目標。如此一來，當您可以執行所需的驗證時，系統就會通知您。如需詳細資訊，請參閱[針對執行個體重新整理事件建立 EventBridge 規則](#)。

考量事項

使用檢查點時，請謹記以下幾點考量：

- 檢查點是以百分比為基礎，因此要取代的執行個體數量會隨著群組的大小而變更。當橫向擴展活動發生且群組的大小增加時，進行中的操作可能會再次到達檢查點。如果發生這種情況，Amazon EC2 Auto Scaling 會傳送另一個通知，並在繼續之前重複檢查點之間的等待時間。
- 在某些情況下，可以略過檢查點。例如，假設 Auto Scaling 群組有兩個執行個體，且檢查點百分比為 [10, 40, 100]。第一個執行個體遭取代後，Amazon EC2 Auto Scaling 會計算 50% 的群組已遭取代。因為 50% 高於前兩個檢查點，其會略過第一個檢查點 (10)，並傳送第二個檢查點 (40) 的通知。
- 取消操作會停止進行任何進一步的取代。如果取消操作或在到達最後一個檢查點前失敗，任何已遭取代的執行個體都不會回復至先前的組態。
- 在部分重新整理的情況下，當您重新執行操作時，Amazon EC2 Auto Scaling 不會從最後一個檢查點重新啟動，也不會僅在較舊執行個體遭取代時停止。但是，在將目標鎖定於新的執行個體之前，其會先針對較舊執行個體進行取代。
- 當檢查點的百分比相對於群組中的執行個體數量太低時，完成的實際百分比可能高於該檢查點的百分比。例如，假設檢查點的百分比為 20%，而群組有四個執行個體。如果 Amazon EC2 Auto Scaling 取代了四個執行個體中的其中一個，則取代的實際百分比 (25%) 將高於檢查點的百分比 (20%)。

- 到達檢查點後，顯示的整體百分比在執行個體完成暖機後才會更新。例如，您的檢查點百分比 [20, 50] 的檢查點延遲為 15 分鐘，運作狀態最低為 80%。您的 Auto Scaling 群組有 10 個執行個體，並進行下列取代：
 - 0:00：用兩個新執行個體取代兩個較舊執行個體。
 - 0:10：兩個新執行個體完成暖機。
 - 0:25：用兩個新執行個體取代兩個較舊執行個體。(為了維持狀態良好最低百分比，只會取代兩個執行個體。)
 - 0:35：兩個新執行個體完成暖機。
 - 0:35：用一個新執行個體取代一個較舊執行個體。
 - 0:45：一個新執行個體完成暖機。

0:35 時，操作停止啟動新執行個體。由於新執行個體沒有完成暖機，完成百分比未準確反映已完成取代的數量 (50%)。新執行個體在 0:45 完成暖機期後，完成百分比會顯示 50%。

- 如果啟用多個檢查點，執行個體重新整理檢查點必須設定為 100%，以確保完全取代所有執行個體。如果您將最終檢查點設定為低於 100% 的百分比（例如 50%），執行個體重新整理會在達到該檢查點百分比後停止取代執行個體，而且不會自動繼續 100% 完成。

使用 AWS 管理主控台 或 啟用檢查點 AWS CLI

您可以使用 AWS 管理主控台 或 AWS CLI 來啟用檢查點。

啟用檢查點 (主控台)

您可以在啟動執行個體重新整理之前啟用檢查點，以使用增量或分階段方法取代執行個體。這會為驗證提供額外的時間。

啟動使用檢查點的執行個體重新整理

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Instance refresh (執行個體重新整理) 索引標籤的 Instance refreshes (執行個體重新整理) 中，選擇 Start instance refresh (啟動執行個體重新整理)。
4. 在 Start instance refresh (啟動執行個體重新整理) 頁面中，輸入 Minimum healthy percentage (運作狀態良好最低百分比) 和 Instance warmup (執行個體暖機期) 的值。

5. 勾選 Enable checkpoints (啟用檢查點) 核取方塊。

這會顯示一個方塊，讓您在其中定義第一個檢查點的百分比閾值。

6. 針對 Proceed until ____ % of the group is refreshed (繼續操作，直到 ____% 的群組經重新整理)，輸入一個數字 (1—100)。這會設定第一個檢查點的百分比。
7. 若要新增其他檢查點，請選擇 Add checkpoint (新增檢查點)，然後定義下一個檢查點的百分比。
8. 若要指定到達檢查點後 Amazon EC2 Auto Scaling 等待的時長，請更新 Wait for **1 hour** between checkpoints (在檢查點之間等待 1 小時) 中的欄位。時間單位可以是小時、分鐘或秒。
9. 如果您已完成處理執行個體重新整理的選取項目，請選擇啟動執行個體重新整理。

啟用檢查點 (AWS CLI)

若要使用 啟動啟用檢查點的執行個體重新整理 AWS CLI，您需要定義下列參數的組態檔案：

- CheckpointPercentages：指定要取代的執行個體百分比閾值。這些閾值提供檢查點。當被取代並暖機的執行個體百分比達到其中一個指定的閾值時，此操作會等待一段指定的時間。您在 CheckpointDelay 中指定要等待的秒數。在指定的時間段過後，執行個體重新整理會繼續執行，直到到達下一個檢查點 (如果適用) 為止。
- CheckpointDelay：指定到達檢查點後再繼續前的等待時間 (以秒為單位)。選擇提供足夠時間來執行驗證的時段。

需要成功取代的 Auto Scaling 群組百分比會以 CheckpointPercentages 陣列中所示的最後一個數值表示。成功取代此百分比之後，且每個執行個體被視為已完成初始化後，該操作會轉換為 Successful。

建立多個檢查點

若要建立多個檢查點，請使用下列範例 [start-instance-refresh](#) 命令。此範例設定了一次執行個體重新整理，這會開始重新整理 Auto Scaling 群組的 1%。等待 10 分鐘後，它會重新整理接下來的 19% 並再等待 10 分鐘。最後，它會在完成操作之前重新整理群組的剩餘部分。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容：

```
{
  "AutoScalingGroupName": "my-asg",
```

```
"Preferences": {
  "InstanceWarmup": 60,
  "MinHealthyPercentage": 80,
  "CheckpointPercentages": [1,20,100],
  "CheckpointDelay": 600
}
```

建立單一檢查點

若要建立單一檢查點，請使用下列範例 [start-instance-refresh](#) 命令。此範例設定了一次執行個體重新整理，這會開始重新整理 Auto Scaling 群組的 20%。等待 10 分鐘後，它會在結束操作之前重新整理群組的剩餘部分。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容：

```
{
  "AutoScalingGroupName": "my-asg",
  "Preferences": {
    "InstanceWarmup": 60,
    "MinHealthyPercentage": 80,
    "CheckpointPercentages": [20,100],
    "CheckpointDelay": 600
  }
}
```

部分重新整理 Auto Scaling 群組

若僅要取代一部分的 Auto Scaling 群組，然後完全停止，請使用下列範例 [start-instance-refresh](#) 命令。此範例設定了一次執行個體重新整理，這會開始重新整理 Auto Scaling 群組的 1%。等待 10 分鐘後，它會在結束操作之前重新整理接下來的 19%。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的內容：

```
{
  "AutoScalingGroupName": "my-asg",
```

```
"Preferences": {
  "InstanceWarmup": 60,
  "MinHealthyPercentage": 80,
  "CheckpointPercentages": [1,20],
  "CheckpointDelay": 600
}
```

根據執行個體生命週期上限來取代 Auto Scaling 執行個體

執行個體生命週期上限會指定一個執行個體在終止和被取代前的服務時間上限 (以秒為單位)。一個常見的使用案例可能是由於內部安全政策或外部合規控制而要求按排程取代您的執行個體。

您必須指定至少 86,400 秒 (一天) 的值。若要清除先前設定的數值，請指定新數值為 0。此設定適用於 Auto Scaling 群組中所有目前和未來的執行個體。

目錄

- [考量事項](#)
- [設定執行個體生命週期上限](#)
- [限制](#)

考量事項

以下是使用此功能時的考量事項：

- 每次取代較舊執行個體並啟動新執行個體時，新執行個體都會使用目前與 Auto Scaling 群組相關聯的啟動範本或啟動組態。如果您的啟動範本或啟動組態指定應用程式不同版本的 Amazon Machine Image (AMI) ID，則會自動部署此版本的應用程式。
- 設定過低的執行個體存留期上限可能會導致執行個體取代速度超過預期。Amazon EC2 Auto Scaling 通常會一次取代一個執行個體，並在取代之間暫停。不過，如果指定的執行個體生命週期上限無法提供足夠的時間個別取代每個執行個體，Amazon EC2 Auto Scaling 必須一次取代多個執行個體。一次取代多個執行個體，最高取代 Auto Scaling 群組目前容量的 10%。為了避免一次取代太多執行個體，請設定較長的執行個體生命週期上限，或使用執行個體縮減保護來暫時防止個別執行個體終止。如需詳細資訊，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。
- Amazon EC2 Auto Scaling 預設會建立新的擴展活動來終止執行個體，然後將其終止。在該執行個體終止時，另一個擴展活動會啟動新的執行個體。您可以使用執行個體維護政策，將此行為變更為在終止前啟動新的執行個體。如需詳細資訊，請參閱[執行個體維護政策](#)。

設定執行個體生命週期上限

在主控台中建立 Auto Scaling 群組時，無法設定執行個體生命週期上限。但是，建立群組後，您可以對其進行編輯來設定執行個體生命週期上限。

設定群組的執行個體生命週期上限 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格，顯示您所選群組的相關資訊。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Advanced configurations (進階組態)、Edit (編輯)。
4. 對於 Maximum instance lifetime (執行個體生命週期上限)，輸入執行個體服務期間的秒數上限。
5. 選擇更新。

在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 下，您可以檢視群組在其歷史記錄中執行個體的取代情況。

設定群組的執行個體生命週期上限 (AWS CLI)

您也可以使用 AWS CLI 為新的或現有的 Auto Scaling 群組設定執行個體生命週期上限。

若要建立新的 Auto Scaling 群組，請使用 [create-auto-scaling-group](#) 命令。

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

以下是 config.json 檔案的範例，顯示執行個體生命週期上限為 2592000 秒 (30 天)。

```
{
  "AutoScalingGroupName": "my-asg",
  "LaunchTemplate": {
    "LaunchTemplateName": "my-launch-template",
    "Version": "$Default"
  },
  "MinSize": 1,
  "MaxSize": 5,
  "MaxInstanceLifetime": 2592000,
```

```
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",  
"Tags": []  
}
```

若要更新現有 Auto Scaling 群組，請使用 [update-auto-scaling-group](#) 命令。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-existing-asg --  
max-instance-lifetime 2592000
```

確認 Auto Scaling 群組的執行個體生命週期上限

使用 [describe-auto-scaling-groups](#) 命令。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

限制

- 不能保證每個執行個體的生命週期上限都是精確的：執行個體不保證只會在最長持續時間結束時被取代。在某些情況下，Amazon EC2 Auto Scaling 可能需要在更新執行個體生命週期上限的參數後立即開始取代執行個體。這種行為的原因是避免同時取代所有執行個體。
- 遵守執行個體縮減保護：Amazon EC2 Auto Scaling 提供執行個體縮減保護，協助您控制可以終止的執行個體。在執行個體上啟用此保護時，即使執行個體已達到其最大執行個體生命週期，Amazon EC2 Auto Scaling 也不會終止執行個體。
- 在啟動前終止的執行個體：當 Auto Scaling 群組中只有一個執行個體時，執行個體生命週期上限功能可能會導致中斷，因為 Amazon EC2 Auto Scaling 會終止執行個體，然後預設啟動新的執行個體。若要將此行為變更為在終止前啟動，請參閱 [執行個體維護政策](#)。

透過擴展增加或減少應用程式的運算容量

擴展能夠增加或減少應用程式的運算容量。擴展會從事件或擴展動作開始，這會指示 Auto Scaling 群組去啟動或終止 Amazon EC2 執行個體。

Amazon EC2 Auto Scaling 提供多種方式來調整擴展，以滿足應用程式的需求。因此，您必須充分了解您的應用程式。請謹記以下幾點考量：

- Amazon EC2 Auto Scaling 在應用程式架構中應該扮演什麼角色？自動擴展通常主要被視為是增加或減少容量的一種方式，然而它對於維持穩定的伺服器數量也很有用。
- 什麼樣的成本限制對您很重要？由於 Amazon EC2 Auto Scaling 使用 EC2 執行個體，您只需為您使用的資源付費。了解您的成本限制，可讓我們協助您決定該何時擴展您的應用程式及其擴展程度。
- 什麼樣的指標對您的應用程式很重要？Amazon CloudWatch 支援多種不同的指標，可讓您與 Auto Scaling 群組搭配使用。

目錄

- [選擇您的擴展方法](#)
- [設定 Auto Scaling 群組的擴展限制](#)
- [設定 Auto Scaling 群組的預設執行個體暖機期](#)
- [Amazon EC2 Auto Scaling 手動擴展](#)
- [Amazon EC2 Auto Scaling 排程擴展](#)
- [Amazon EC2 Auto Scaling 動態擴展](#)
- [Amazon EC2 Auto Scaling 的預測擴展](#)
- [控制縮減期間應終止的 Auto Scaling 執行個體](#)
- [暫停和繼續 Amazon EC2 Auto Scaling 程序](#)

選擇您的擴展方法

Amazon EC2 Auto Scaling 為您提供了數種方式來擴展 Auto Scaling 群組。

維持固定數量的執行個體

Auto Scaling 群組預設沒有任何連接的擴展政策或排程動作，這使它維持固定的大小。在您建立 Auto Scaling 群組之後，透過啟動足夠的執行個體來滿足其所需容量，它就會開始運作。如果群組沒有連接擴展條件，即使執行個體運作狀態不佳，該群組仍會繼續維持所需容量。Amazon EC2 Auto Scaling

會監控 Auto Scaling 群組中每個執行個體的運作狀態。當它發現某個執行個體的運作狀態不佳時，就會用新的執行個體取代該執行個體。您可以在 [Auto Scaling 群組中的執行個體運作狀態檢查](#) 中閱讀關於此過程的更深入的描述。

手動擴展

手動擴展是最基本的 Auto Scaling 群組擴展方式。您可以更新 Auto Scaling 群組的所需容量，或終止 Auto Scaling 群組中的執行個體。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 手動擴展](#)。

依據排程擴展

依排程擴展意味著擴展動作會自動執行為日期和時間的函數。當您確切知道何時要增加或減少群組中執行個體的數量時，這一點非常有用，這完全是因為需求是按照可預測的時程而產生的。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 排程擴展](#)。

根據需求動態擴展

使用動態擴展功能來擴展資源的更進階方式可讓您定義擴展政策，透過動態方式重新調整 Auto Scaling 群組來符合需求變更。例如，假設您目前有一個 Web 應用程式在兩個執行個體上執行，而且您希望當應用程式的負載變更時，Auto Scaling 群組的 CPU 使用率保持在 50% 左右。當您不知道流量何時變更時，此方法對於隨著流量變更進行擴展很有用。您可以設定擴展政策為您做出回應。您可以使用多種政策類型（或它們的組合）來擴展，以回應流量變更。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 動態擴展](#)。

主動擴展

您也可以結合預測擴展和動態擴展（分別為主動和被動方法）更快地擴展 EC2 容量。在每日和每週流量模式之前，使用預測擴展來增加 Auto Scaling 群組中的 EC2 執行個體數量。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 的預測擴展](#)。

設定 Auto Scaling 群組的擴展限制

擴展限制代表您的 Auto Scaling 群組所需的最小和最大群組大小。您可以分別為最小和最大大小設定限制值。

群組的所需容量可以調整為最小和最大大小限制範圍內的數字。預期容量必須等於或大於群組大小下限，且等於或小於群組大小上限。

- **所需容量：**代表建立之時的 Auto Scaling 群組的初始容量。Auto Scaling 群組會嘗試維持所需容量。它透過啟動所需容量的指定執行個體數量而開始，只要 Auto Scaling 群組沒有連接擴展政策或排程動作，便會維持此執行個體數量。

- 容量下限：代表群組大小下限。設定擴展政策時，這些政策不能將群組的所需容量減少到低於容量下限。
- 容量上限：代表群組大小上限。設定擴展政策時，這些政策不能將群組的所需容量增加到高於容量上限。

大小限制下限和上限在下列情況也適用：

- 當您透過更新其所需容量，手動擴展 Auto Scaling 群組時。
- 當更新所需容量的排程動作執行時。如果排程的動作在執行時未為群組指定新的大小限制下限和上限，則群組的當前大小限制的下限和上限適用。

Auto Scaling 群組會始終嘗試維持其所需容量。如果執行個體意外終止 (例如，由於 Spot 執行個體中斷、運作狀態檢查失敗或人為動作)，群組會自動啟動新執行個體以維持其所需容量。

在主控台中管理這些設定

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在導覽窗格的 Auto Scaling 下，選擇 Auto Scaling Groups (Auto Scaling 群組)。
3. 在 Auto Scaling groups (Auto Scaling 群組) 頁面中，選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在下窗格中的詳細資訊索引標籤上，檢視或變更目前設定的群組所需容量、容量下限和容量上限。如需詳細資訊，請參閱[更改現有 Auto Scaling 群組的所需容量](#)。

在詳細資訊窗格上方，您可找到相關資訊，例如 Auto Scaling 群組中目前的執行個體數量、所需容量、容量下限和容量上限，以及一個狀態欄。如果 Auto Scaling 群組使用執行個體權重，您也可以找到對所需容量有所貢獻的容量單位數量。

若要新增或移除清單中的欄位，請選擇頁面頂端的設定圖示。接著在 Auto Scaling groups attributes (Auto Scaling 群組屬性) 中，開啟或關閉每一欄位，然後選擇 Confirm (確認)。

變更後驗證 Auto Scaling 群組的大小

Instances (執行個體) 欄顯示目前正在執行的執行個體數量。啟動或終止執行個體時，Status (狀態) 欄會顯示 Updating capacity (正在更新容量) 狀態，如下圖所示。

Auto Scaling groups (1/1)							
🔄 Edit Delete Create an Auto Scaling group							
🔍 my-asg ✕ 1 match < 1 > ⚙️							
☒	Name ▾	Launch template... ▾	Instances ▾	Status ▾	Desired... ▾	Min ▾	Max ▲
☒	my-asg	my_template Version Def	0	⋮ Updating capacity	1	0	1

請稍候幾分鐘，然後重新整理檢視，以查看最新狀態。擴展活動完成後，Instances (執行個體) 欄會顯示更新後的值。

您可以從 Instances (執行個體) 的 Instance management (執行個體管理) 索引標籤下檢視執行個體的數量，以及目前正在執行的執行個體的状态。

設定 Auto Scaling 群組的預設執行個體暖機期

CloudWatch 在您的 Auto Scaling 執行個體中收集和彙總使用資料，例如 CPU 和網路輸入/輸出。您可以使用這些指標來建立擴展政策，而這些政策會根據所選指標值的增加和減少來調整 Auto Scaling 群組中的執行個體數量。

您可以指定執行個體達到其在將用量資料提供給彙總指標之前等待 InService 狀態的時間。此指定時間稱為預設執行個體暖機期。這可避免動態擴展受到尚未處理應用程式流量之個別執行個體的指標影響，並且可能暫時遇到高使用量的運算資源。

為了最佳化目標追蹤和步進擴展政策的效能，強烈建議您啟用和設定預設執行個體暖機期。預設不會啟用或設定。

當您啟用預設執行個體暖機期時，請記住，如果您的 Auto Scaling 群組設定為使用執行個體維護政策，或者您使用執行個體重新整理來取代執行個體，您可以防止執行個體在完成初始化之前計入運作狀態最低百分比。

目錄

- [擴展效能考量](#)
- [選擇預設執行個體暖機時間](#)
- [為群組啟用預設執行個體暖機期](#)
- [驗證群組的預設執行個體暖機期](#)
- [尋找具有先前設定之執行個體暖機時間的擴展政策](#)
- [清除先前為擴展政策設定的執行個體暖機期](#)

擴展效能考量

對於大多數應用程式來說，有一個預設的執行個體暖機時間適用於所有功能，而不是不同功能的不同暖機時間是有用的。例如，如果您未設定預設執行個體暖機期，執行個體重新整理功能會使用運作狀態檢查寬限期做為預設暖機期。如果您有任何目標追蹤和步進擴展政策，它們會使用為預設冷卻時間設定的值作為預設暖機時間。如果您有任何預測擴展政策，則沒有預設的暖機時間。

當執行個體暖機時，只有在未暖機的執行個體指標值大於政策的警示高閾值（或目標追蹤擴展政策的目標使用率）時，您的動態擴展政策才會向外擴展。如果需求減少，動態擴展會變得更保守，以保護應用程式的可用性。這會封鎖動態擴展的縮減活動，直到新的執行個體完成暖機。

向外擴展時，Amazon EC2 Auto Scaling 會在決定要新增至群組的執行個體數量時，將暖機的執行個體視為群組容量的一部分。因此，需要新增類似容量的多個警示違規會導致單一擴展活動。目的是持續向外擴展，而不會過度擴展。

如果未啟用預設執行個體暖機期，則執行個體在將指標傳送至 CloudWatch 並將其計入目前容量之前等待的時間會因執行個體而異。因此，與實際發生的工作負載相比，擴展政策有可能無法預測地執行。

例如，可以假設一個具有重複間歇工作負載模式的應用程式。預測擴展政策可用於針對是否增加執行個體數量做出週期性決策。由於預測擴展政策沒有預設的暖機時間，執行個體會立即開始對彙總指標做出貢獻。如果這些執行個體在啟動時有較高的資源使用量，則新增執行個體可能會導致彙總指標激增。這可能會影響使用這些指標的任何動態擴展政策，視使用量穩定下來所需的時間而定。如果違反動態擴展政策的高警示閾值，則群組的大小會再次增加。當新的執行個體暖機時，縮減活動將會遭到封鎖。

選擇預設執行個體暖機時間

設定預設執行個體暖機期的關鍵在於確定執行個體完成初始化所需的時間，以及在執行個體達到 InService 狀態後資源消耗穩定下來所需的時間。選擇執行個體暖機時間時，請嘗試在收集合法流量的用量資料，以及將與啟動時臨時用量峰值相關聯的資料收集降到最低之間保持最佳平衡。

假設您有一個 Auto Scaling 群組連接到 Elastic Load Balancing 負載平衡器。新的執行個體完成啟動後，它們會註冊到負載平衡器，然後再進入 InService 狀態。在執行個體進入 InService 狀態後，資源耗用仍然會遇到暫時峰值，並需要時間來穩定。例如，相較於無需下載大型資產的輕量型 Web 伺服器，必須下載和快取大型資產的應用程式伺服器的資源耗用需要更長的穩定時間。執行個體暖機期提供了穩定資源耗用所需的時間延遲。

Important

如果您不確定需要多少時間進行暖機，可以從 300 秒開始。然後逐漸減少或增加，直到您取得應用程式的最佳擴展效能為止。您可能需要這樣做幾次才能正確完成。或者，如果您有任何具

有自己的暖機時間 (EstimatedInstanceWarmup) 的擴展政策，您可以使用此值開始。如需詳細資訊，請參閱[尋找具有先前設定之執行個體暖機時間的擴展政策](#)。

可以考慮將生命週期關聯用於在啟動時要執行組態任務或指令碼的使用案例。生命週期關聯可以延遲新的執行個體投入使用，直到其完成初始化。它們特別適用於您的自舉指令碼需要一段時間才能完成的情況。如果您新增 lifecycle hook，則可以減少預設執行個體暖機期的值。如需有關 lifecycle hook 的詳細資訊，請參閱 [Amazon EC2 Auto Scaling lifecycle hook](#)。

為群組啟用預設執行個體暖機期

您可以在建立 Auto Scaling 群組時啟用預設執行個體暖機期。您也可以為現有的群組啟用此功能。

透過啟用預設執行個體暖機功能，您不再需要為下列功能的暖機參數指定值：

- [執行個體重新整理](#)
- [目標追蹤擴展](#)
- [步進擴展](#)

Console

為新群組啟用預設執行個體暖機期 (主控台)

當您建立 Auto Scaling 群組時，於 Configure advanced options (設定進階選項) 頁面的 Additional settings (其他設定) 中，選取 Enable default instance warmup (啟用預設執行個體暖機期) 選項。選擇應用程式所需的暖機時間。

AWS CLI

為新群組啟用預設執行個體暖機期 (AWS CLI)

要為 Auto Scaling 群組啟用預設執行個體暖機期，請新增 `--default-instance-warmup` 選項並指定一個值 (以秒為單位)，範圍從 0 到 3600。啟用後，-1 的值將關閉此設定。

以下 [create-auto-scaling-group](#) 命令會建立一個名為 `my-asg` 的 Auto Scaling 群組，並啟用值為 `120` 秒的預設執行個體暖機期。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --  
default-instance-warmup 120 ...
```

 Tip

如果此命令擲回錯誤，請確定您已在 AWS CLI 本機將 更新為最新版本。

Console

為現有群組啟用預設執行個體暖機期 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所在的 AWS 區域。
3. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在 Details (詳細資訊) 索引標籤上，選擇 Advanced configurations (進階組態)、Edit (編輯)。
5. 針對預設執行個體暖機期，選擇應用程式所需的暖機期。
6. 選擇更新。

AWS CLI

為現有群組啟用預設執行個體暖機期 (AWS CLI)

以下範例使用 [update-auto-scaling-group](#) 命令名為 *my-asg* 的現有 Auto Scaling 群組啟用預設執行個體暖機期，且其值為 *120* 秒。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --  
default-instance-warmup 120
```

 Tip

如果此命令擲回錯誤，請確定您已在 AWS CLI 本機將 更新為最新版本。

驗證群組的預設執行個體暖機期

使用以下程序，使用 驗證 Auto Scaling 群組的預設執行個體暖機期 AWS CLI。

驗證 Auto Scaling 群組的預設執行個體暖機期

使用以下 [describe-auto-scaling-groups](#) 命令。將 *my-asg* 取代為您的 Auto Scaling 群組名稱。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下是回應範例。

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn",
      ...
      "DefaultInstanceWarmup": 120
    }
  ]
}
```

尋找具有先前設定之執行個體暖機時間的擴展政策

若要識別您的政策是否有自己的 暖機時間 `EstimatedInstanceWarmup`，請使用 執行下列 [describe-policies](#) 命令 AWS CLI。將 *my-asg* 取代為您的 Auto Scaling 群組名稱。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
--query 'ScalingPolicies[?EstimatedInstanceWarmup!=`null`]'
```

以下為範例輸出。

```
[
  {
    "AutoScalingGroupName": "my-asg",
    "PolicyName": "cpu50-target-tracking-scaling-policy",
    "PolicyARN": "arn",
    "PolicyType": "TargetTrackingScaling",
    "StepAdjustments": [],
    "EstimatedInstanceWarmup": 120,
    "Alarms": [{
      "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
    }
  ]
}
```



```

        "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
    },
    {
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
        "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }
],
"TargetTrackingConfiguration":{
    "PredefinedMetricSpecification":{
        "PredefinedMetricType":"ASGAverageCPUUtilization"
    },
    "TargetValue":50.0,
    "DisableScaleIn":false
},
"Enabled":true
},
... additional policies ...

]

```

清除先前為擴展政策設定的執行個體暖機期

啟用預設執行個體暖機後，請更新任何仍有自己的暖機時間的擴展政策，以清除先前設定的值。否則，它將覆寫預設的執行個體暖機期。

您可以使用 主控台 AWS CLI 或 AWS SDKs 更新擴展政策。本章節介紹使用主控台的步驟。如果您使用 AWS CLI AWS SDKs，請確定您保留現有的政策組態，但移除 `EstimatedInstanceWarmup` 屬性。當您更新現有的擴展政策時，政策將被您以程式設計方式呼叫 [PutScalingPolicy](#) 時指定的內容取代。不會保留原始值。

清除先前為擴展政策設定的執行個體暖機期 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。
頁面底部會開啟一個分割窗格。
3. 在自動擴展索引標籤的動態擴展政策中，選擇您感興趣的政策，然後選擇動作、編輯。

4. 對於執行個體暖機期，請清除執行個體暖機期值，以改用預設執行個體暖機期值。
5. 選擇更新。

Amazon EC2 Auto Scaling 手動擴展

您可以隨時手動調整 Auto Scaling 群組中的 EC2 執行個體數量。手動變更執行個體計數的程序稱為手動擴展。手動擴展是自動擴展的替代方案，特別是如果您想要進行一次性容量變更時。

手動擴展群組之後，Amazon EC2 Auto Scaling 會根據您定義的擴展政策和排程動作，繼續正常的自動擴展活動。對於已啟用預設執行個體暖機期的群組，任何新執行個體都會經過暖機期，然後才會開始影響用於自動擴展的指標。此暖機期有助於以新容量穩定群組。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

有時，您可能需要暫時停用擴展政策和排程動作，然後再手動擴展群組。這樣做可以防止手動擴展動作和自動擴展活動之間產生衝突。如需詳細資訊，請參閱[關閉擴展活動](#)。

目錄

- [更改現有 Auto Scaling 群組的所需容量](#)
- [終止 Auto Scaling 群組中的執行個體 \(AWS CLI\)](#)

更改現有 Auto Scaling 群組的所需容量

當您變更 Auto Scaling 群組的所需容量時，Amazon EC2 Auto Scaling 會管理啟動和終止執行個體的程序，以達到新的所需大小。

Console

變更 Auto Scaling 群組的大小

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

分割窗格會顯示在頁面底部。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Group details (群組詳細資訊)、Edit (編輯)。
4. 對於所需容量，增加或減少所需容量。例如，若要將群組的大小增加一，如果目前值為 1，請輸入 2。

如果所需容量的新值大於所需容量下限和所需容量上限，則所需容量上限會自動增加到新的所需容量值。

5. 完成後，選擇 Update (更新)。

確認您指定的群組大小導致啟動相同數量的執行個體。例如，如果您將群組的大小增加一個，請確認 Auto Scaling 群組已啟動一個額外的執行個體。

驗證 Auto Scaling 群組大小已變更

1. 在活動索引標籤的活動歷史記錄中，您可以檢視與 Auto Scaling 群組相關聯的活動進度。Status (狀態) 欄顯示執行個體的目前狀態。在執行個體啟動的同時，狀態欄顯示 Not yet in service。在執行個體啟動後，狀態會變更成 Successful。您也可以使用重新整理圖示來查看執行個體的目前狀態。如需詳細資訊，請參閱[驗證 Auto Scaling 群組的擴展活動](#)。
2. 在執行個體管理索引標籤的執行個體中，您可以檢視執行個體的状态。啟動執行個體無須費時。
 - Lifecycle (生命週期) 欄顯示執行個體的状态。初始時，您的執行個體為 Pending 状态。在執行個體準備好接受流量之後，它的状态會變成 InService。
 - 運作状态欄顯示執行個體上 Amazon EC2 Auto Scaling 運作状态檢查的結果。

AWS CLI

以下範例假設您已建立 Auto Scaling 群組，其大小上限為 5，下限為 1。所以，群組目前有一個執行中的執行個體。

變更 Auto Scaling 群組的大小

使用 [set-desired-capacity](#) 命令變更 Auto Scaling 群組的大小，如以下範例所示。

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
  --desired-capacity 2
```

如果選擇遵守 Auto Scaling 群組的預設冷卻時間，您必須如以下範例所示指定 `--honor-cooldown` 選項。如需詳細資訊，請參閱[擴展 Amazon EC2 Auto Scaling 冷卻時間](#)。

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
```

```
--desired-capacity 2 --honor-cooldown
```

驗證 Auto Scaling 群組大小

使用 [describe-auto-scaling-groups](#) 命令確認 Auto Scaling 群組的大小已變更，如以下範例所示。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下是輸出範例，提供啟動的群組和執行個體的詳細資訊。

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn",
      "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
      },
      "MinSize": 1,
      "MaxSize": 5,
      "DesiredCapacity": 2,
      "DefaultCooldown": 300,
      "AvailabilityZones": [
        "us-west-2a"
      ],
      "LoadBalancerNames": [],
      "TargetGroupARNs": [],
      "HealthCheckType": "EC2",
      "HealthCheckGracePeriod": 300,
      "Instances": [
        {
          "ProtectedFromScaleIn": false,
          "AvailabilityZone": "us-west-2a",
          "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1",
            "LaunchTemplateId": "lt-050555ad16a3f9c7f"
          },
          "InstanceId": "i-05b4f7d5be44822a6",
          "InstanceType": "t3.micro",
          "HealthStatus": "Healthy",
```

```

        "LifecycleState": "Pending"
      },
      {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1",
          "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-0c20ac468fa3049e8",
        "InstanceType": "t3.micro",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
      }
    ],
    "CreatedTime": "2019-03-18T23:30:42.611Z",
    "SuspendedProcesses": [],
    "VPCZoneIdentifier": "subnet-c87f2be0",
    "EnabledMetrics": [],
    "Tags": [],
    "TerminationPolicies": [
      "Default"
    ],
    "NewInstancesProtectedFromScaleIn": false,
    "ServiceLinkedRoleARN": "arn",
    "TrafficSources": []
  }
]
}

```

請注意，DesiredCapacity 顯示新的數值。您的 Auto Scaling 群組已啟動一個額外的執行個體。

終止 Auto Scaling 群組中的執行個體 (AWS CLI)

有時您可能想要在 Auto Scaling 群組中手動縮減，但又想要終止特定的執行個體。您可以透過使用 [terminate-instance-in-auto-scaling-group](#) 命令，並指定要終止的執行個體的 ID 和 `--should-decrement-desired-capacity` 選項，以在 Auto Scaling 群組中手動縮減，如下列範例所示。

```
aws autoscaling terminate-instance-in-auto-scaling-group \
  --instance-id i-026e4c9f62c3e448c --should-decrement-desired-capacity
```

以下是輸出範例，提供擴展活動的詳細資訊。

```
{
  "Activities": [
    {
      "ActivityId": "b8d62b03-10d8-9df4-7377-e464ab6bd0cb",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-026e4c9f62c3e448c",
      "Cause": "At 2023-09-23T06:39:59Z instance i-026e4c9f62c3e448c was taken out of service in response to a user request, shrinking the capacity from 1 to 0.",
      "StartTime": "2023-09-23T06:39:59.015000+00:00",
      "StatusCode": "InProgress",
      "Progress": 0,
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":\"us-west-2c\"}"
    }
  ]
}
```

此選項在主控台中無法使用。不過，您可以使用 Amazon EC2 主控台的執行個體頁面來終止 Auto Scaling 群組中的執行個體。當您這麼做時，Amazon EC2 Auto Scaling 會偵測到執行個體不再執行，並自動將其取代為運作狀態檢查程序的一部分。在新執行個體啟動之前，終止執行個體需要一到兩分鐘的時間。如需有關如何終止執行個體的資訊，請參閱《Amazon EC2 使用者指南》中的[終止執行個體](#)。

如果您終止 群組中的執行個體，導致跨可用區域分佈不均，Amazon EC2 Auto Scaling 會重新平衡群組以重新建立均勻分佈，除非您暫停AZRebalance程序。如需詳細資訊，請參閱[暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。

Amazon EC2 Auto Scaling 排程擴展

透過排程擴展，您可以根據可預測的負載變更為應用程式設定自動擴展。您可以建立排程動作，在特定時間增加或減少群組所需的容量。

例如，您會遇到固定的每週流量模式，其中負載會在一週中增加，並在一週結束時下降。您可以在 Amazon EC2 Auto Scaling 中設定符合此模式的擴展排程：

- 在星期三上午，一個排程動作會透過增加先前設定的 Auto Scaling 群組所需容量來增加容量。
- 在星期五晚上，另一個排程動作會透過減少先前設定的 Auto Scaling 群組所需容量來減少容量。

藉由這些排定的擴展動作，您可將費用和效能調整到最佳狀態。您的應用程式有足夠的容量來處理週中流量峰值，但在其他時間不會過度佈建不需要的容量。

您可以同時使用排程擴展和擴展政策，以獲得這兩種擴展方法的優勢。執行排定的擴展動作後，擴展政策可以繼續決定是否進一步擴展容量。這有助於確保您有足夠的容量來處理應用程式的負載。當應用程式擴展以滿足需求時，目前的容量必須落在您排定的動作所設定的容量上下限之內。

目錄

- [排程擴展的運作方式](#)
- [週期性排程](#)
- [時區](#)
- [考量事項](#)
- [限制](#)
- [建立排程動作](#)
- [檢視排程動作詳細資訊](#)
- [刪除排程動作](#)

排程擴展的運作方式

若要使用排程擴展，請建立排程動作，指示 Amazon EC2 Auto Scaling 在特定時間執行擴展活動。當您建立排程動作時，您可以指定 Auto Scaling 群組、應進行擴展活動的時間、新的所需容量，以及選擇性的新最小容量和新的最大容量。您可以建立僅擴展一次或依週期性排程擴展的排程動作。

在指定的時間，Amazon EC2 Auto Scaling 會比較目前容量與指定的所需容量，以根據新的容量值進行擴展。

- 如果目前容量小於指定的所需容量，Amazon EC2 Auto Scaling 會橫向擴展或新增執行個體至指定的所需容量。
- 如果目前容量大於指定的所需容量，Amazon EC2 Auto Scaling 會將執行個體縮減或移除為指定的所需容量。

排程動作會在指定的日期和時間設定群組的所需容量、容量下限和容量上限。您一次只能為其中一個容量建立排程動作，例如所需的容量。不過，在某些情況下，您必須包含最小和最大容量，以確保您在動作中指定的所需容量不超出這些限制。

週期性排程

若要使用 AWS CLI 或 SDK 建立週期性排程，請指定 cron 表達式和時區來描述該排程動作何時重複。您可以選擇性地為開始時間、結束時間 (或兩者) 指定日期和時間。

若要使用 建立週期性排程 AWS 管理主控台，請指定排程動作的週期模式、時區、開始時間和選用結束時間。所有週期性模式選項都是以 Cron 表達式為基礎。您也可以撰寫自己的自訂 Cron 表達式。

受支援的 Cron 表達式格式由 5 個以空格分隔的欄位組成：[分鐘] [小時] [一個月的第幾日] [一年的第幾個月] [一週的第幾日]。例如，Cron 表達式 `30 6 * * 2` 會設定排程動作，每週二上午 6:30 重複執行。使用星號作為萬用字元，以比對欄位的所有數值。如需 Cron 表達式的其他範例，請參閱 <https://crontab.guru/examples.html>。如需有關以此格式撰寫自己的 Cron 表達式的相關資訊，請參閱 [Crontab](#)。

請審慎選擇開始與結束時間。請謹記以下幾點：

- 如果指定開始時間，Amazon EC2 Auto Scaling 會在此時執行動作，然後會根據指定週期執行該動作。
- 如果指定了結束時間，過了此時刻會停止此動作。排程動作一旦達到結束時間，將不會存留於您的帳戶中。
- 如果週期時間完全符合結束時間，Amazon EC2 Auto Scaling 將不會在結束時間執行排程動作。
- 當您使用 AWS CLI 或 SDK 時，開始時間和結束時間必須以 UTC 設定。

時區

依預設，您設定的週期性排程會使用國際標準時間 (UTC)。您可以變更時區以對應至您當地的時區或網路另一個部分的時區。在指定遵循日光節約時間 (DST) 的時區時，動作會自動針對 DST 進行調整。

有效值是網際網路指派號碼授權機構 (IANA) 時區資料庫中時區的正式名稱。例如，美國東部時間正式識別為 `America/New_York`。如需詳細資訊，請參閱 <https://www.iana.org/time-zones>。

位置型時區，例如 `America/New_York` 會自動調整 DST。不過，以 UTC 為基礎的時區 (例如 `Etc/UTC`) 是絕對時間，不會針對 DST 進行調整。

例如，您有時區為 `America/New_York` 的週期性排程。第一個擴展動作會發生在 DST 啟動之前的 `America/New_York` 時區。下一個擴展動作會發生在 DST 啟動之後的 `America/New_York` 時區。第一個動作會在當地時間 UTC-5 上午 8:00 開始，而第二個動作則會在當地時間 UTC-4 上午 8:00 開始。

如果您使用 建立排程動作，AWS 管理主控台 並指定觀察 DST 的時區，週期性排程和開始和結束時間都會自動調整 DST。

考量事項

當您建立排程動作時，請謹記下列事項：

- 適用於排程動作的執行命令在同一群組內會受保證，但跨群組的排程動作則否。
- 排程動作通常會在幾秒鐘內執行。不過，動作可能會比排定的開始時間延遲兩分鐘。由於 Auto Scaling 群組內的排程動作會依指定的順序來執行，因此排定的開始時間彼此接近的動作可能會執行更長的時間。
- 您可以暫停 ScheduledActions 程序，暫時關閉 Auto Scaling 群組的排程擴展。這可協助您避免排程動作處於作用中狀態，而不需要將其刪除。然後，您可以在想要再次使用時繼續執行排程擴展。如需詳細資訊，請參閱 [暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。
- 建立排程動作後，您可以更新其名稱以外的任何設定。
- 當相同 Auto Scaling 群組中的多個排程動作具有相同的 Cron 表達式時，執行順序會變成任意且未定義。為了確保可預測的行為，您可以為每個排程動作使用唯一的排程開始時間。

限制

- 每個 Auto Scaling 群組的排程動作名稱必須是唯一名稱。
- 排程動作都必須有唯一的時間值。如果嘗試在已排程另一個擴展活動的時間排程活動，則呼叫會遭拒並傳回錯誤，指出已存在此排程開始時間的排程動作。
- 每個 Auto Scaling 群組最多可以建立 125 個排程動作。

建立排程動作

若要為 Auto Scaling 群組建立排程動作，請使用下列其中一種方法：

Console

建立排程動作

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Automatic scaling (自動擴展) 索引標籤的 Scheduled actions (排程動作) 中，選擇 Create scheduled action (建立排程動作)。
4. 輸入排程動作的 Name (名稱)。
5. 在所需容量、容量下限、容量上限下，選擇群組的新的所需容量，以及新的容量下限和容量上限。預期容量必須等於或大於群組大小下限，且等於或小於群組大小上限。
6. 針對 Recurrence (週期)，選擇其中一個可用選項。
 - 如果想要依週期性排程進行擴展，請選擇 Amazon EC2 Auto Scaling 應執行排程動作的頻率。
 - 如果選擇以 Every (每個) 開始的選項，將為您建立 Cron 表達式。
 - 如果選擇 Cron，請輸入指定何時執行動作的 Cron 表達式。
 - 如果只想縮放一次，請選擇 Once (一次)。
7. 對於 Time zone (時區)，選擇時區。預設值為 Etc/UTC。

所有列出的時區都來自 IANA 時區資料庫。如需詳細資訊，請參閱 https://en.wikipedia.org/wiki/List_of_tz_database_time_zones。

8. 在 Specific start time (特定開始時間) 中定義日期和時間。
 - 如果選擇週期性排程，開始時間會定義週期性序列中第一個排程作業的執行時間。
 - 如果選擇 Once (一次) 作為週期性設定，則開始時間會定義排程動作執行的日期和時間。
9. (選用) 對於週期性排程，您可以選擇 Set End Time (設定結束時間)，然後為 End by (結束時間) 選擇日期和時間來指定結束時間。
10. 選擇建立。主控台會顯示 Auto Scaling 群組的排程動作。

AWS CLI

若要建立排程動作，您可以使用下列其中一個範例命令。將每個#####替換為自己的資訊。

範例：僅擴展一次

使用下列 [put-scheduled-update-group-action](#) 命令搭配 --start-time "YYYY-MM-DDThh:mm:ssZ"和 --desired-capacity選項。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-action \
```

```
--auto-scaling-group-name my-asg --start-time "2021-03-31T08:00:00Z" --desired-capacity 3
```

範例：依據週期性排程排定擴展

使用下列 [put-scheduled-update-group-action](#) 命令搭配 `--recurrence "cron expression"` 和 `--desired-capacity` 選項。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \  
--auto-scaling-group-name my-asg --recurrence "0 9 * * *" --desired-capacity 3
```

根據預設，Amazon EC2 Auto Scaling 會根據 UTC 時區執行指定的週期排程。若要指定不同的時區，請包含 `--time-zone` 選項與 IANA 時區的名稱，如下列範例所示。

```
--time-zone "America/New_York"
```

如需詳細資訊，請參閱 https://en.wikipedia.org/wiki/List_of_tz_database_time_zones。

檢視排程動作詳細資訊

若要檢視 Auto Scaling 群組即將排程動作的詳細資訊，請使用下列其中一種方法：

Console

檢視排程動作詳細資訊

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組。
3. 在自動擴展索引標籤的排程動作區段中，您可以檢視即將到來的排程動作。

請注意，主控台會顯示本機時間中開始時間和結束時間的值，其中 UTC 偏移在指定的日期和時間生效。UTC 時差是從本地時間到 UTC 的差異，以小時和分鐘為單位。Time zone (時區) 的值顯示您所要求的時區，例如 `America/New_York`。

AWS CLI

使用下列 [describe-scheduled-actions](#) 命令。

```
aws autoscaling describe-scheduled-actions --auto-scaling-group-name my-asg
```

如果成功，此命令傳回的輸出會類似如下。

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2020-12-01T00:30:00Z",
      "Time": "2020-12-01T00:30:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4
    }
  ]
}
```

驗證擴展活動

若要驗證與排程擴展相關聯的擴展活動，請參閱 [驗證 Auto Scaling 群組的擴展活動](#)。

刪除排程動作

若要刪除排程動作，請使用下列其中一種方法：

Console

刪除排程動作

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組。
3. 在 Automatic scaling (自動擴展) 索引標籤的 Scheduled actions (排程動作) 中，選取排程動作。
4. 選擇 動作、刪除。

5. 出現確認提示時，選擇 Yes, Delete (是，刪除)。

AWS CLI

使用下列 [delete-scheduled-action](#) 命令。

```
aws autoscaling delete-scheduled-action --auto-scaling-group-name my-asg \  
--scheduled-action-name my-recurring-action
```

Amazon EC2 Auto Scaling 動態擴展

動態擴展會隨著流量變化而擴展 Auto Scaling 群組的容量。

Amazon EC2 Auto Scaling 支援下列動態擴展政策類型：

- 目標追蹤擴展 - 根據 Amazon CloudWatch 指標和目標值，增加和減少目前的群組容量。這與家中溫控器維持溫度的方式類似：您只要選擇溫度，然後溫控器就會完成其他的動作。
- 步進擴展 - 根據一組擴展值來增加和減少群組目前的容量，也稱為步進調整，會根據警示違規的程度而有不同。
- 簡易擴展 - 根據單一擴展調整增加和減少群組的目前容量，每個擴展活動之間有一個冷卻時間。

我們強烈建議您使用目標追蹤擴展政策，並選擇與 Auto Scaling 群組容量變更成反向比例的指標。因此，如果您將 Auto Scaling 群組的大小加倍，指標會減少 50%。如此一來，指標資料就能依比例準確觸發擴展事件。包含指標，例如平均 CPU 使用率或每個目標的平均請求計數。

透過目標追蹤，Auto Scaling 群組會根據應用程式的實際負載，以直接比例擴展。這意味著除了滿足應對負載變化的即時容量需求外，目標追蹤政策還可以適應隨時間而發生的負載變化，例如由於季節性變化。

目標追蹤政策也不需要手動定義 CloudWatch 警示和擴展調整。Amazon EC2 Auto Scaling 會根據您設定的目標自動處理此問題。

目錄

- [動態擴展政策的運作方式](#)
- [多個動態擴展政策](#)
- [Amazon EC2 Auto Scaling 的目標追蹤擴展政策](#)
- [Amazon EC2 Auto Scaling 政策的步進和簡易擴展政策](#)

- [擴展 Amazon EC2 Auto Scaling 冷卻時間](#)
- [以 Amazon SQS 為基礎的擴展政策](#)
- [驗證 Auto Scaling 群組的擴展活動](#)
- [停用 Auto Scaling 群組的擴展政策](#)
- [刪除 Auto Scaling 群組的擴展政策](#)
- [的範例擴展政策 AWS CLI](#)

動態擴展政策的運作方式

動態擴展政策指示 Amazon EC2 Auto Scaling 追蹤特定的 CloudWatch 指標，並定義當相關 CloudWatch 警示處於 ALARM 狀態時要採取的動作。用於呼叫警示狀態的指標是來自 Auto Scaling 群組中所有執行個體的指標彙總。(例如，假設您有一個 Auto Scaling 群組具有兩個執行個體，其中一個執行個體的 CPU 使用率是 60%，另一個 CPU 使用率是 40%。平均而言，其 CPU 使用率是 50%)。政策生效後，Amazon EC2 Auto Scaling 會在超出警示閾值時，向上或向下調整群組的所需容量。

叫用動態擴展政策時，如果容量計算產生的數字超過群組的大小下限和上限範圍，Amazon EC2 Auto Scaling 會確保新容量永遠不會超過大小限制的下限和上限。容量的測量方式有兩種：使用您在執行個體設定所需容量時選擇的相同單位，或使用容量單位（如果套用[執行個體權重](#)）。

- 範例 1：Auto Scaling 群組的容量上限為 3，目前容量為 2，以及增加 3 個執行個體的動態擴展政策。叫用此政策時，Amazon EC2 Auto Scaling 只會將 1 個執行個體新增至群組，防止群組超過其大小上限。
- 範例 2：Auto Scaling 群組的容量下限為 2，目前容量為 3，以及移除 2 個執行個體的動態擴展政策。叫用此政策時，Amazon EC2 Auto Scaling 只會從群組移除 1 個執行個體，防止群組變成小於其大小下限。

當所需容量達到大小上限時，會停止擴增。如果需求下降，而且容量減少，則 Amazon EC2 Auto Scaling 可以再次進行擴增。

例外狀況是當您使用執行個體權重時。在這種情況下，Amazon EC2 Auto Scaling 可以擴增到大小上限以上，但最多只能擴展到您的最大執行個體權重。其目的是盡可能接近新的所需容量，但仍會遵循為該群組指定的配置策略。配置策略會決定要啟動的執行個體類型。權重會根據其執行個體類型決定每個執行個體對群組所需容量有多少容量單位。

- 範例 3：Auto Scaling 群組的容量上限為 12，目前容量為 10，以及增加 5 個容量單位的動態擴展政策。執行個體類型已指派以下三個權重之一：1、4 或 6。叫用此政策時，Amazon EC2 Auto Scaling

會根據配置策略選擇啟動權重為 6 的執行個體類型。此擴增事件的結果是所需容量為 12 且目前容量為 16 的群組。

多個動態擴展政策

在大部分情況下，目標追蹤擴展政策已足夠設定 Auto Scaling 群組自動擴增或縮減。目標追蹤擴展政策可讓您選擇所需的結果，並令 Auto Scaling 群組視需要新增和移除執行個體，以達成該結果。

對於進階擴展組態而言，Auto Scaling 群組可以擁有多個擴展政策。例如，您可以定義一或多個目標追蹤擴展政策、一或多個步進擴展政策，或兩者兼具。這樣可以提供更大的靈活性，以涵蓋多種情境。

為了說明多個動態擴展政策如何搭配使用，請考慮採用一個使用 Auto Scaling 群組的應用程式，以及一個將請求傳送給群組中單一 EC2 執行個體的 Amazon SQS 佇列。為了確認應用程式在最佳狀態等級下執行，有兩種政策控制 Auto Scaling 群組何時該執行擴增。一個是目標追蹤政策，使用自訂指標，依據佇列中 SQS 訊息的數量新增和移除容量。另一個則是步進擴展政策，它會在執行個體的使用率在一指定時間長度內超過 90% 時使用 Amazon CloudWatch CPUUtilization 指標來增加容量。

當多個政策同時生效時，有可能每一個政策同時都會指示 Auto Scaling 群組擴增 (或縮減)。例如，在 SQS 自訂指標達到尖峰並超出自訂指標警示的閾值時，CPUUtilization 指標也可能同時達到尖峰並超出 CloudWatch 警示的閾值。

發生這些情況時，Amazon EC2 Auto Scaling 會選擇對於擴增和縮減均可提供容量上限的政策。例如，假設 CPUUtilization 政策啟動一個執行個體，SQS 佇列政策啟動兩個執行個體。如果擴增條件同時滿足這兩個政策，則 Amazon EC2 Auto Scaling 會優先採用 SQS 佇列政策。這樣會使 Auto Scaling 群組啟動兩個執行個體。

即使各政策使用不同的縮減條件，仍然適用優先選擇能提供容量上限的政策。例如，如果一個政策終止三個執行個體，另一個政策減少 25% 的執行個體數，且群組在縮減時有八個執行個體，Amazon EC2 Auto Scaling 會優先選擇為該群組提供最大數量執行個體的政策。這樣會使 Auto Scaling 群組終止兩個執行個體 (8 的 25% = 2)。這種做法的目的是防止 Amazon EC2 Auto Scaling 移除過多的執行個體。

不過，我們建議您小心使用目標追蹤擴展政策與步進擴展政策，因為這些政策之間的衝突可能會造成非預期行為。例如，如果步驟擴展政策在目標追蹤政策準備好縮減之前啟動縮減活動，則縮減活動將不會遭到封鎖。縮減活動完成後，目標追蹤政策可以指示群組再次向外擴展。

Amazon EC2 Auto Scaling 的目標追蹤擴展政策

目標追蹤擴展政策會根據目標指標值自動擴展 Auto Scaling 群組的容量。它會自動適應個別應用程式的獨特使用模式。這可讓您的應用程式維持 EC2 執行個體的最佳效能和高使用率，以提高成本效益，而無需手動介入。

如果使用目標追蹤，您必須選取指標和目標值，以代表應用程式理想的平均使用率或輸送量。Amazon EC2 Auto Scaling 會建立和管理 CloudWatch 警示，在指標偏離目標時調用擴展事件。例如，這類似於調溫器如何維持目標溫度。

例如，假設您目前有一個應用程式在兩個執行個體上執行，而且您希望當應用程式的負載變更時，Auto Scaling 群組的 CPU 使用率保持在 50% 左右。這樣可以給予您額外的容量處理流量尖峰，而不會維持過多的閒置資源數。

您可以透過建立以 50% 的平均 CPU 利用率為目標的目標追蹤擴展政策來滿足此需求。然後，當 CPU 超過 50% 來處理增加的負載時，Auto Scaling 群組將橫向擴展或增加容量。當 CPU 低於 50% 時，它會縮減或減少容量，以在低使用率期間最佳化成本。

主題

- [多個目標追蹤擴展政策](#)
- [選擇 Metrics \(指標\)](#)
- [定義目標值](#)
- [定義執行個體暖機時間](#)
- [考量事項](#)
- [建立目標追蹤擴展政策](#)
- [使用高解析度指標建立目標追蹤政策，以加快回應速度](#)
- [使用指標數學建立目標追蹤擴展政策](#)

多個目標追蹤擴展政策

為協助最佳化擴展效能，您可使用多個目標追蹤擴展政策，但前提是各政策使用不同的指標。例如，使用率和輸送量可能會相互影響。每當這些指標之一發生變化時，通常意味著其他指標也會受到影響。因此，使用多個指標可提供有關 Auto Scaling 群組所承受負載的其他資訊。這有助於 Amazon EC2 Auto Scaling 在判斷要新增至群組的容量時，做出更明智的決策。

Amazon EC2 Auto Scaling 的目的是一律優先考慮可用性。如果任何目標追蹤政策已準備好向外擴展，則會向外擴展 Auto Scaling 群組。只有在所有目標追蹤政策（已啟用縮減部分）都準備好縮減時，才會縮減。

選擇 Metrics (指標)

您可以使用預先定義的指標或自訂指標建立目標追蹤擴展政策。預先定義的指標可讓您更輕鬆地存取最常用於擴展的指標。自訂指標可讓您擴展其他可用的 CloudWatch 指標，包括[以幾秒鐘的間隔更精細發佈的高解析度指標](#)。您可以發佈自己的高解析度指標或其他服務發佈的 AWS 指標。

如需使用高解析度指標建立目標追蹤政策的詳細資訊，請參閱[使用高解析度指標建立目標追蹤政策，以加快回應速度](#)。

目標追蹤支援下列預先定義的指標：

- `ASGAverageCPUUtilization`：Auto Scaling 群組的 CPU 平均使用率。
- `ASGAverageNetworkIn`：Auto Scaling 群組在所有網路介面上收到的平均位元組數量。
- `ASGAverageNetworkOut`：Auto Scaling 群組在所有網路介面上傳送出去的平均位元組數量。
- `ALBRequestCountPerTarget` — Auto Scaling 群組每個目標的平均 Application Load Balancer 請求計數。

Important

有關每個目標的 CPU 使用率、網路 I/O 和 Application Load Balancer 請求計數指標的其他寶貴資訊，請參閱《Amazon EC2 使用者指南》中的[列出執行個體主題的可用 CloudWatch 指標](#)，以及《Application [Application Load Balancer](#) 使用者指南》中的 [Application Load Balancer 主題的 CloudWatch 指標](#)。

您可以在 CloudWatch 中透過指定自訂指標，選擇其他可用的 CloudWatch 指標或自己的指標。如需使用為目標追蹤擴展政策指定自訂指標規格的範例 AWS CLI，請參閱[的範例擴展政策 AWS CLI](#)。

選擇指標時請謹記以下事項：

- 我們建議您只使用一分鐘或更低間隔可用的指標，以協助您更快地擴展以回應使用率變更。以較低間隔發佈的指標可讓目標追蹤政策更快地偵測和回應 Auto Scaling 群組使用率的變化。

- 如果您選擇 Amazon EC2 發佈的預先定義指標，例如 CPU 使用率，建議您啟用詳細監控。根據預設，所有 Amazon EC2 指標會以五分鐘的間隔發佈，但可透過啟用詳細監控，將其設定為較低的一分鐘間隔。如需如何啟用詳細監控的資訊，請參閱 [設定 Auto Scaling 執行個體的監控](#)。
- 並非所有的自訂指標都適用於目標追蹤。指標必須是有效的使用率指標，而且能夠表示執行個體的忙碌程度。指標值必須與 Auto Scaling 群組中的執行個體數成比例增加或減少。如此，指標資料便可依比例依執行個體數量擴增或縮減。例如，如果 Auto Scaling 群組的負載分佈於各執行個體之間，則 Auto Scaling 群組的 CPU 使用率是有效的 (也就是具有指標維度 `AutoScalingGroupName` 的 Amazon EC2 指標 `CPUUtilization`)。
- 以下指標不適用於目標追蹤：
 - 面對 Auto Scaling 群組的負載平衡器所收到的請求數量 (也就是 Elastic Load Balancing 指標 `RequestCount`)。負載平衡器收到的請求數量不會根據 Auto Scaling 群組的使用率而改變。
 - 負載平衡器請求延遲 (也就是 Elastic Load Balancing 指標 `Latency`)。請求延遲會隨使用率提高而增加，但不一定會依比例變動。
 - CloudWatch Amazon SQS 佇列指標 `ApproximateNumberOfMessagesVisible`。佇列中的訊息數量可能不會隨處理佇列訊息之 Auto Scaling 群組的大小成比例地變更。然而，測量 Auto Scaling 群組中每個 EC2 執行個體佇列中訊息數量的自訂指標是可行的。如需詳細資訊，請參閱 [以 Amazon SQS 為基礎的擴展政策](#)。
- 若要使用 `ALBRequestCountPerTarget` 指標，您必須指定 `ResourceLabel` 參數來識別與指標相關聯的負載平衡器目標群組。如需使用 指定目標追蹤擴展政策 `ResourceLabel` 參數的範例 AWS CLI，請參閱 [的範例擴展政策 AWS CLI](#)。
- 當指標向 CloudWatch 真的發出 0 值時 (例如 `ALBRequestCountPerTarget`)，如果應用程式持續一段時間沒有流量，Auto Scaling 群組可能縮減到 0。若要讓 Auto Scaling 群組在沒有請求傳送到它時縮減為 0，該群組的容量下限必須設定為 0。
- 您可以使用指標數學來合併現有的指標，而不是發布新的指標來用於您的擴展政策。如需詳細資訊，請參閱 [使用指標數學建立目標追蹤擴展政策](#)。

定義目標值

建立目標追蹤擴展政策時，您必須指定目標值。目標值代表 Auto Scaling 群組的最佳平均使用率或輸送量。若要以符合成本效益的方式使用資源，請盡可能高地設定目標值，並為非預期的流量增加提供合理的緩衝區。當您的應用程式進行最佳橫向擴展以達到正常流量時，實際指標值應等於或略低於目標值。

擴展政策以輸送量為基礎時 (例如 Application Load Balancer 每個目標的要求計數、網路 I/O 或其他計數指標)，目標值代表一分鐘期間單一執行個體的最佳平均輸送量。

定義執行個體暖機時間

您可以選擇性指定新啟動的執行個體暖機所需的秒數。在指定的暖機時間結束之前，執行個體不會計入 Auto Scaling 群組的彙總 EC2 執行個體指標。

當執行個體處於暖機期間時，只有在未暖機的執行個體指標值大於政策的目標使用率時，擴展政策才會向外擴展。

如果群組再次水平擴展，則仍在暖機的執行個體將計為下次水平擴展活動所需的容量一部分。這種做法的目的是連續的向外擴展 (但並非過度)。

當向外擴展活動正在進行時，所有由擴展政策啟動的向內擴展活動都會遭到封鎖，直到執行個體完成暖機。當執行個體完成暖機時，如果發生擴展事件，則在計算新的所需容量時，目前正在終止的任何執行個體都會計入群組的目前容量。所以，我們不會從 Auto Scaling 群組移除超過必要數量的執行個體。

預設值

如果未設定任何值，擴展政策將使用預設值，這是為群組定義之[預設執行個體暖機](#)期的值。如果預設執行個體暖機期為 null，則會回到[預設冷卻時間](#)的值。我們建議您使用預設執行個體暖機期，以便在暖機期變更時更輕鬆地更新所有擴展政策。

考量事項

使用目標追蹤擴展政策時，下列考量適用：

- 請勿建立、編輯或刪除用於目標追蹤擴展政策的 CloudWatch 警示。Amazon EC2 Auto Scaling 會建立和管理與您的目標追蹤擴展政策相關聯的 CloudWatch 警示，並視需要編輯、取代或刪除它們，以自訂應用程式的擴展體驗及其不斷變化的使用率模式。
- 目標追蹤擴展政策會在流量減少時逐漸縮減，在流量水平波動期間排定可用性優先順序。如果您想要更好的控制，步驟擴展政策可能是更好的選項。您可以暫時停用目標追蹤政策的縮減部分。這有助於維持成功部署的執行個體數量下限。
- 如果指標缺少資料點，這會導致 CloudWatch 警示狀態變更為 INSUFFICIENT_DATA。發生這種情況時，Amazon EC2 Auto Scaling 無法擴展您的群組，直到找到新的資料點為止。
- 如果指標在設計上是以稀疏的方式來報告，指標數學可能會有所幫助。例如，若要使用最新的值，請使用指標為 m1 的 FILL(m1, REPEAT) 函數。
- 您可能會看到目標值和實際指標資料點之間有些差距。原因我們會在決定新增或移除多少執行個體時，透過向上或向下四捨五入保守地行動。如此一來，防止我們新增不足數量的執行個體，或移除太多的執行個體。然而，對於較小的 Auto Scaling 群組使用較少的執行個體，該群組的使用率可能與目標值相差甚遠。

例如，假設您為 CPU 使用率設定了 50% 的目標值，然後 Auto Scaling 群組就會超過目標。我們可能會判斷新增 1.5 執行個體使 CPU 使用率降低接近 50%。由於不可能新增 1.5 執行個體，我們四捨五入而新增兩個執行個體。這可能會降低 CPU 使用率低於 50%，但可確保有足夠的資源來支援您的應用程式。同樣地，如果我們判斷移除 0.5 個執行個體會將您的 CPU 使用率提高到 50% 以上，我們將選擇不縮減，直到指標降低到我們認為縮減不會造成振盪的程度為止。

對於具有較多執行個體的大型 Auto Scaling 群組，使用率會分佈於較多的執行個體，在這種情況下，新增或移除執行個體所導致目標值和實際指標資料點之間的差距會比較小。

- 目標追蹤擴展政策假設在指定的指標超過目標值時，應擴增您的 Auto Scaling 群組。當指定的指標低於目標值時，您無法使用目標追蹤擴展政策來橫向擴展 Auto Scaling 群組。

建立目標追蹤擴展政策

若要為您的 Auto Scaling 群組建立目標追蹤擴展政策，請使用下列其中一種方法。

開始之前，請確認您偏好的指標每隔 1 分鐘可用 (相比之下，Amazon EC2 指標的預設間隔為 5 分鐘)。

Console

建立新的 Auto Scaling 群組的目標追蹤擴展政策

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。
3. 在步驟 1、2 和 3 中，視需要選擇選項，並繼續進行 Step 4: Configure group size and scaling policies (步驟 4：設定群組大小和擴展政策)。
4. 在擴展下，透過更新所需容量下限和所需容量上限，指定要在兩者之間擴展的範圍。這兩個設定可讓您的 Auto Scaling 群組進行動態擴展。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
5. 在自動擴展下，選擇目標追蹤擴展政策。
6. 若要定義政策，請執行下列的動作：
 - a. 指定政策的名稱。
 - b. 為 Metric type (指標類型) 選擇指標。

如果選擇了 Application Load Balancer request count per target (每個目標的 Application Load Balancer 請求計數)，則在 Target group (目標群組) 中選擇目標群組。

- c. 指定指標的 Target value (目標值)。
- d. (選用) 對於執行個體暖機期，視需要更新執行個體暖機期值。
- e. (選用) 選取 Disable scale in to create only a scale-out policy (停用縮減以僅建立擴增政策)。這可讓您建立所需不同類型的單獨縮減政策。

7. 繼續建立 Auto Scaling 群組。您的擴展政策會在建立 Auto Scaling 群組之後建立。

建立現有 Auto Scaling 群組的目標追蹤擴展政策

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 確認已將擴展限制設定妥當。例如，如果群組已達到所需容量上限，則需要指定新的上限，以便進行橫向擴展。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
4. 在 Automatic scaling (自動擴展) 索引標籤的 Dynamic scaling policies (動態擴展政策) 中，選擇 Create dynamic scaling policy (建立動態擴展政策)。
5. 若要定義政策，請執行下列的動作：
 - a. 對於政策類型，請保持目標追蹤擴展的預設值。
 - b. 指定政策的名稱。
 - c. 為 Metric type (指標類型) 選擇指標。您只能選擇一種指標類型。若要使用多個指標，請建立多個政策。

如果選擇了 Application Load Balancer request count per target (每個目標的 Application Load Balancer 請求計數)，則在 Target group (目標群組) 中選擇目標群組。

- d. 指定指標的 Target value (目標值)。
- e. (選用) 對於執行個體暖機期，視需要更新執行個體暖機期值。
- f. (選用) 選取 Disable scale in to create only a scale-out policy (停用縮減以僅建立擴增政策)。這可讓您建立所需不同類型的單獨縮減政策。

6. 選擇建立。

AWS CLI

若要建立目標追蹤擴展政策，您可以使用下列範例來協助您開始使用。將每個#####替換為自己的資訊。

Note

如需更多範例，請參閱[範例擴展政策 AWS CLI](#)。

建立目標追蹤擴展政策 (AWS CLI)

1. 使用下列cat命令，將擴展政策的目標值和預先定義的指標規格存放在主目錄中名為config.json的JSON檔案中。以下是將平均CPU使用率保持在50%的目標追蹤組態範例。

```
$ cat ~/config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "ASGAverageCPUUtilization"
    }
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的[PredefinedMetricSpecification](#)。

2. 使用 [put-scaling-policy](#) 命令與先前步驟中建立的 config.json 檔案來建立您的擴展政策。

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

如果成功，此命令會傳回代表您建立的兩個 CloudWatch 警示的 ARN 和名稱。

```
{
  "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
```

```

    aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/cpu50-target-tracking-
    scaling-policy",
    "Alarms": [
      {
        "AlarmARN": "arn:aws:cloudwatch:us-
        west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
        fc0e4183-23ac-497e-9992-691c9980c38e",
        "AlarmName": "TargetTracking-my-asg-AlarmHigh-
        fc0e4183-23ac-497e-9992-691c9980c38e"
      },
      {
        "AlarmARN": "arn:aws:cloudwatch:us-
        west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
        bd9e-471a352ee1a2",
        "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
        bd9e-471a352ee1a2"
      }
    ]
  }
}

```

使用高解析度指標建立目標追蹤政策，以加快回應速度

目標追蹤支援高解析度 CloudWatch 指標，並以低於一分鐘的間隔發佈秒級資料點。設定目標追蹤政策，透過高解析度 CloudWatch 指標監控具有波動需求模式的應用程式使用率，例如用戶端服務 APIs、即時串流服務、電子商務網站和隨需資料處理。為了在符合容量與需求時達到更高的精確度，目標追蹤會使用這種精細的監控，更快速地偵測和回應不斷變化的 EC2 執行個體需求和使用率。

如需如何以高解析度發佈指標的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[發佈自訂指標](#)。若要存取和發佈 EC2 指標，例如高解析度 CPU 使用率，您可能想要使用 [CloudWatch 代理程式](#)。

AWS 區域

除了 AWS 區域 之外，所有 都可以使用高解析度指標進行目標追蹤 AWS GovCloud (US) Regions。

具有高解析度指標的目標追蹤政策如何運作

您可以透過定義要追蹤的指標和要為指標維護的目標值，來建立目標追蹤政策。若要擴展高解析度指標，請指定指標的名稱，並將目標追蹤觀察此指標的指標期間設定為低於 60 秒的值。目前支援的最低間隔為 10 秒。您可以比此更低的間隔發佈指標。

Note

不支援大於 60 的指標期間。

您可以在單一 CloudWatch 指標上設定目標追蹤，或查詢多個 CloudWatch 指標，並使用數學表達式根據這些指標建立新的單一時間序列。這兩個選項都可讓您定義指標期間。

範例**範例 1**

下列範例會根據高解析度 CloudWatch 指標建立目標追蹤政策。指標會以 10 秒解析度發佈。透過定義期間，您可以啟用目標追蹤，以 10 秒精細程度監控此指標。將每個#####替換為自己的資訊。

```
$ cat ~/config.json
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
    "MetricName": "MyHighResolutionMetric",
    "Namespace": "MyNamespace",
    "Dimensions": [
      {
        "Name": "MyOptionalDimensionName",
        "Value": "MyOptionalMetricDimensionValue"
      }
    ],
    "Statistic": "Average",
    "Unit": "None"
    "Period": "10"
  }
}
```

範例 2

您可以使用指標數學表達式，將多個指標合併為單一時間序列以進行擴展。指標數學在將現有指標轉換為平均每個執行個體時特別有用。轉換指標至關重要，因為目標追蹤假設指標與 Auto Scaling 群組的容量成反比。因此，當容量增加時，指標應該會減少近乎相同的比例。

例如，假設您的指標代表應用程式要處理的待處理任務。您可以使用指標數學，將待處理任務除以 Auto Scaling 群組的執行容量。Auto Scaling 會以 1 分鐘的精細程度發佈容量指標，因此此指標在一分鐘間

隔內不會有任何值。如果您想要使用更高的解析度進行擴展，這可能會導致容量與待定任務指標之間的期間不相符。為了避免此不相符，建議您使用 FILL 表達式，以上一分鐘時間戳記中記錄的容量編號填入缺少的值。

下列範例使用指標數學將待定任務指標除以容量。在這段期間，我們會將這兩個指標設定為 10 秒。由於指標是以 1 分鐘的間隔發佈，因此我們在容量指標上使用 FILL 操作。

使用指標數學來修改多個指標

```
{
  "CustomizedMetricSpecification": {
    "Metrics": [
      {
        "Label": "Pending jobs to be processed",
        "Id": "m1",
        "MetricStat": {
          "Metric": {
            "MetricName": "MyPendingJobsMetric",
            "Namespace": "Custom",
          },
          "Stat": "Sum",
          "Period": 10
        },
        "ReturnData": false
      },
      {
        "Label": "Get the running instance capacity (matching the period to
that of the m1)",
        "Id": "m2",
        "MetricStat": {
          "Metric": {
            "MetricName": "GroupInServiceInstances",
            "Namespace": "AWS/AutoScaling",
            "Dimensions": [
              {
                "Name": "AutoScalingGroupName",
                "Value": "my-asg"
              }
            ]
          },
          "Stat": "Average",
          "Period": 10
        }
      },
    ]
  }
}
```



```

        "ReturnData": false
    },
    {
        "Label": "Calculate the pending job per capacity (note the use of the
FILL expression)",
        "Id": "e1",
        "Expression": "m1 / FILL(m2,REPEAT)",
        "ReturnData": true
    }
]
},
"TargetValue": 100
}

```

考量事項

使用目標追蹤和高解析度指標時，請考慮下列事項。

- 為了確保您沒有可能導致不需要的自動擴展結果的遺失資料點，您的 CloudWatch 指標必須以與您指定期間相同的或更高解析度發佈。
- 將目標值定義為您要為 Auto Scaling 群組維護 per-instance-per-minute 指標值。如果您使用的指標值可以根據指標的期間進行乘以，則設定適當的目標值至關重要。例如，任何以計數為基礎的指標，例如使用 SUM 統計資料的請求計數或待定任務，都會有不同的指標值，取決於所選的期間。您仍應假設您要根據每分鐘平均值設定目標。
- 雖然使用 Amazon EC2 Auto Scaling 無需額外費用，但您必須支付資源，例如 Amazon EC2 執行個體、CloudWatch 指標和 CloudWatch 警示。上述範例中建立的高解析度警示定價與標準 CloudWatch 警示不同。如需 CloudWatch 定價的詳細資訊，請參閱 [Amazon CloudWatch 定價](#)。
- 目標追蹤需要指標代表 EC2 執行個體的每個執行個體平均使用率。若要達成此目的，您可以使用 [指標數學操作](#) 做為目標追蹤政策組態的一部分。將指標除以 Auto Scaling 群組的執行容量。請務必為您用來建立單一時間序列的每個指標定義相同的指標期間。如果這些指標以不同的間隔發佈，請在具有較高間隔的指標上使用 FILL 操作來填入遺失的資料點。

使用指標數學建立目標追蹤擴展政策

利用指標數學，可查詢多個 CloudWatch 指標，並使用數學表達式根據這些指標來建立新的時間序列。您可以在 CloudWatch 主控台視覺化產生的時間序列，並將其新增至儀表板。如需有關指標數學的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [使用指標數學](#)。

指標數學運算式有下列考量：

- 您可以查詢任何可用的 CloudWatch 指標。每個指標都是指標名稱、命名空間以及零個或多個維度的唯一組合。
- 您可以使用任何算術運算子 (+-*/^)、統計函數 (如 AVG 或 SUM) 或 CloudWatch 支援的其他函數。
- 您可以在數學運算式的公式中同時使用其他數學運算式的指標和結果。
- 在指標規範中使用的任何運算式都必須最終傳回單一的時間序列。
- 您可以透過使用 CloudWatch 主控台或 CloudWatch [GetMetricData](#) API 驗證指標數學表達式是否有效。

範例：每個執行個體的 Amazon SQS 佇列待辦項目

若要計算每個執行個體的 Amazon SQS 佇列待辦項目，請獲取佇列中可擷取的大致訊息數量，然後將該數字除以 Auto Scaling 群組的執行中容量 (即處於 InService 狀態的執行個體數量)。如需詳細資訊，請參閱[以 Amazon SQS 為基礎的擴展政策](#)。

運算式的邏輯如下所示：

$\text{sum of (number of messages in the queue) / (number of InService instances)}$

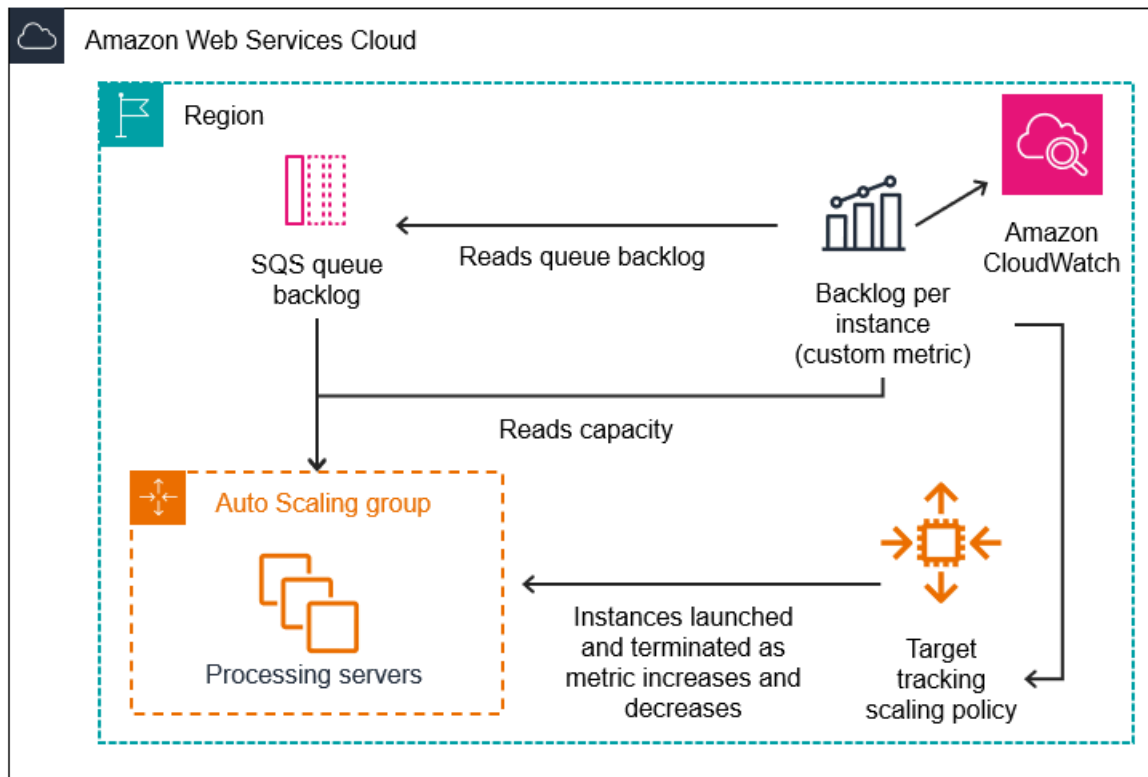
那麼，您的 CloudWatch 指標資訊如下。

ID	CloudWatch 指標	統計數字	Period
m1	ApproximateNumberOfMessagesVisible	總和	1 分鐘
m2	GroupInServiceInstances	平均數	1 分鐘

您的指標數學 ID 和表達式如下。

ID	表達式
e1	$(m1)/(m2)$

下圖說明此指標的架構：



使用此指標數學建立目標追蹤擴展政策 (AWS CLI)

1. 將指標數學運算式作為自訂指標規格的一部分儲存在名為 `config.json` 的 JSON 檔案中。

使用以下範例協助您開始使用。將每個 `#####` 替換為自己的資訊。

```
{
  "CustomizedMetricSpecification": {
    "Metrics": [
      {
        "Label": "Get the queue size (the number of messages waiting to be processed)",
        "Id": "m1",
        "MetricStat": {
          "Metric": {
            "MetricName": "ApproximateNumberOfMessagesVisible",
            "Namespace": "AWS/SQS",
            "Dimensions": [
              {
                "Name": "QueueName",
                "Value": "my-queue"
              }
            ]
          }
        }
      }
    ]
  }
}
```

```

        ]
      },
      "Stat": "Sum"
    },
    "ReturnData": false
  },
  {
    "Label": "Get the group size (the number of InService instances)",
    "Id": "m2",
    "MetricStat": {
      "Metric": {
        "MetricName": "GroupInServiceInstances",
        "Namespace": "AWS/AutoScaling",
        "Dimensions": [
          {
            "Name": "AutoScalingGroupName",
            "Value": "my-asg"
          }
        ]
      },
      "Stat": "Average"
    },
    "ReturnData": false
  },
  {
    "Label": "Calculate the backlog per instance",
    "Id": "e1",
    "Expression": "m1 / m2",
    "ReturnData": true
  }
]
},
"TargetValue": 100
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [TargetTrackingConfiguration](#)。

Note

以下是一些其他資源，可協助您尋找 CloudWatch 指標的指標名稱、命名空間、維度和統計資料：

- 如需 AWS 服務的可用指標相關資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [發佈 CloudWatch 指標的 AWS 服務](#)。
- 要使用 AWS CLI 取得 CloudWatch 指標的確切指標名稱、命名空間和維度 (如果適用)，請參閱 [list-metrics](#)。

2. 若要建立此政策，執行使用 JSON 檔案作為輸入 [put-scaling-policy](#) 命令，如以下範例所示。

```
aws autoscaling put-scaling-policy --policy-name sqs-backlog-target-tracking-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \  
  --target-tracking-configuration file://config.json
```

如果成功，則此命令會傳回政策的 Amazon Resource Name (ARN) 和代表您建立的兩個 CloudWatch 警示的 ARN。

```
{  
  "PolicyARN": "arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-  
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/sqs-backlog-target-  
tracking-scaling-policy",  
  "Alarms": [  
    {  
      "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-  
fc0e4183-23ac-497e-9992-691c9980c38e",  
      "AlarmName": "TargetTracking-my-asg-AlarmHigh-  
fc0e4183-23ac-497e-9992-691c9980c38e"  
    },  
    {  
      "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-  
bd9e-471a352ee1a2",  
      "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-  
bd9e-471a352ee1a2"  
    }  
  ]  
}
```

 Note

如果此命令擲回錯誤，請確定您已在 AWS CLI 本機將 更新為最新版本。

Amazon EC2 Auto Scaling 政策的步進和簡易擴展政策

步驟擴展和簡單擴展政策會根據 CloudWatch 警示，以預先定義的增量來擴展 Auto Scaling 群組的容量。您可以定義個別的擴展政策，在流量達到警示閾值時擴增規模 (增加容量) 和縮減規模 (減少容量)。

如果您使用執行個體**權重**，則會根據執行個體或容量單位來測量 Auto Scaling 群組容量。此外，所需容量和目前容量之間有差異。

- 所需容量 – 您想要在 群組中擁有的執行個體 (或容量單位) 數量。所需的容量可以手動調整，也可以使用擴展政策自動調整。
- 目前容量 – 群組中已超過暖機和冷卻期間，且正在執行並可供使用的執行個體 (或容量單位) 數量。

透過步驟擴展和簡單擴展，您可以建立和管理叫用擴展程序的 CloudWatch 警示。違反警示時，Amazon EC2 Auto Scaling 會啟動與該警示相關聯的擴展政策。

強烈建議使用目標追蹤擴展政策，根據平均 CPU 使用率或每個目標的平均請求計數等指標進行擴展。容量增加時減少和容量減少時增加的指標，可用於使用目標追蹤依比例擴增或縮減執行個體數量。這有助於確保 Amazon EC2 Auto Scaling 緊密地遵循應用程式的需求曲線。如需詳細資訊，請參閱[目標追蹤擴展政策](#)。

目錄

- [步進擴展政策的運作方式](#)
- [步進擴展的步進調整](#)
- [擴展調整類型](#)
- [執行個體暖機期](#)
- [考量事項](#)
- [建立橫向擴展的步驟擴展政策](#)
- [建立向內擴展的步驟擴展政策](#)

- [簡易擴展政策](#)

步進擴展政策的運作方式

若要使用步驟擴展，請先建立 CloudWatch 警示，以監控 Auto Scaling 群組的指標。定義指標、閾值和評估流量是否達到警示閾值的時段數。然後，建立步驟擴展政策，定義如何在超過警示閾值時擴展您的群組。您可以針對擴展調整類型使用 Auto Scaling 群組或容量單位目前容量的百分比。如需詳細資訊，請參閱[擴展調整類型](#)。

在政策中新增步進調整內容。您可以根據流量達到警示閾值的程度，定義不同的步進調整幅度。例如：

- 如果警示指標達到 60%，則橫向擴展 10 個執行個體
- 如果警示指標達到 75%，則橫向擴展 30 個執行個體
- 如果警示指標達到 85%，則橫向擴展 40 個執行個體

在指定的評估期間數目內違反警示閾值時，Amazon EC2 Auto Scaling 會套用政策中定義的步驟調整。警示狀態恢復 OK 後，才能依其他達標警示事件繼續調整。

每個執行個體都有一個暖機期，以防止擴展活動對短時間內發生的變更太過反應。您可以選擇性地設定擴展政策的暖機期。不過，我們建議您使用預設執行個體暖機期，以便在暖機期變更時更輕鬆地更新所有擴展政策。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

簡單擴展政策類似於步進擴展政策，但它們是以單一擴展調整為基礎，每個擴展活動之間都有冷卻時間。如需詳細資訊，請參閱[簡易擴展政策](#)。

步進擴展的步進調整

建立步進擴展政策時，您可以指定一或多個步進調整，這些調整會根據警示違規的程度自動動態調整執行個體數量。每項步進調整可指定下列項目：

- 指標值下限
- 指標值上限
- 要擴展的數量，會以擴展調整類型為依據

CloudWatch 會根據與 CloudWatch 警示相關聯的指標統計資料彙總指標資料點。超出警示閾值時，會呼叫適當的擴展政策。Amazon EC2 Auto Scaling 會將彙總類型套用於來自 CloudWatch 的最近指標資料點 (而不是原始指標資料)。它會將彙總指標值與步進調整定義的上限和下限進行比較，以決定要執行哪一項步進調整。

您可以指定相對於違規閾值的上限及下限。例如，假設您設定流量超過 50% 指標時，由 CloudWatch 發出警示並執行擴增政策。當流量低於 50% 指標時，系統發出第二個警示，並執行縮減政策。您對每個政策進行了一組步驟調整，調整類型為 PercentChangeInCapacity (或 主控台 中的 群組百分比)：

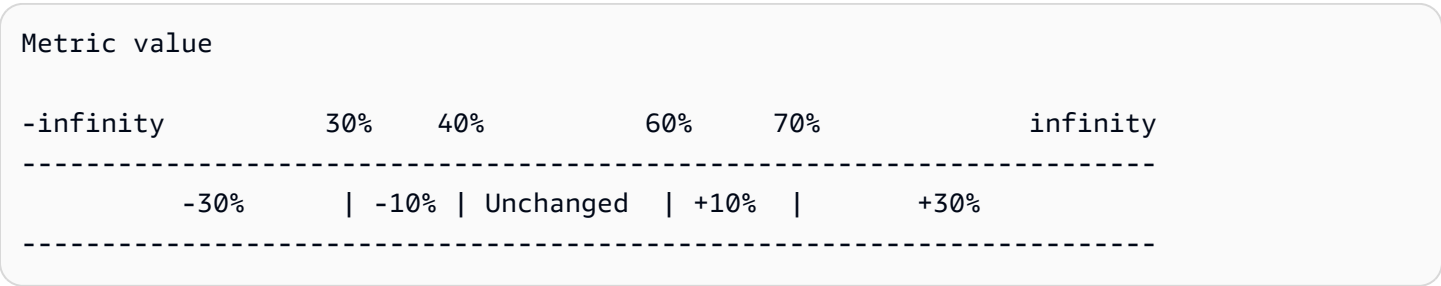
範例：擴增政策的步進調整

下限	上限	調整
0	10	0
10	20	10
20	無	30

範例：縮減政策的步進調整

下限	上限	調整
-10	0	0
-20	-10	-10
null	-20	-30

這會建立以下擴展組態。



現在，假設您在同時具有目前容量和所需容量為 10 的 Auto Scaling 群組上使用此擴展組態。下列幾點摘要說明擴展組態相對於所需容量和目前群組容量的行為：

- 當彙總指標值大於 40 且小於 60 時，將維持所需的容量和目前的容量。

- 如果指標值達到 60，根據擴增政策的第二項步進調整 (增加 10 的執行個體的 10%)，群組所需的容量會增加 1 的執行個體而達到 11 個執行個體。在新執行個體執行且其指定的暖機時間已過期後，群組的目前容量會增加至 11 個執行個體。如果指標值在容量增加後仍增加到 70，則群組的所需容量會再增加 3 個執行個體，達到 14 個執行個體。這是根據擴增政策的第三步調整 (增加 11 個執行個體的 30%，即 3.3 個執行個體，無條件捨去為 3 個執行個體)。
- 如果指標值為 40，根據縮減政策的第二項步進調整 (移除 14 個執行個體的 10%，即 1.4 個執行個體，無條件捨入到 1 個執行個體)，群組所需容量會減少 1 個執行個體而變成 13 個執行個體。如果指標值在容量減少後仍降至 30，則群組的所需容量會再減少 3 個執行個體，變成 10 個執行個體。這是根據縮減政策的第三步調整 (減去 13 個執行個體的 30%，即 3.9 個執行個體，無條件捨去為 3 個執行個體)。

當您為擴展政策指定步進調整時，請注意下列事項：

- 如果您使用 AWS 管理主控台，您可以將上限和下限指定為絕對值。如果您使用 AWS CLI 或 開發套件，您可以指定相對於違規閾值的上限和下限。
- 步進調整的範圍不得重疊或有間隙。
- 僅其中一項步進調整的下限可為空值 (負無限大)。若某項步進調整的下限為負值，則必須有一項步進調整的下限為空值。
- 僅其中一項步進調整的上限可為空值 (正無限大)。若某項步進調整的上限為正值，則必須有一項步進調整的上限為空值。
- 同一項步進調整的上限及下限不得皆為空值。
- 如果指標值高於違規閾值，則含下限而不含上限。如果指標值低於違規閾值，則不含下限而含上限。

擴展調整類型

您可以根據選擇的擴展調整類型，定義執行最佳擴展動作的擴展政策。您可以將調整類型指定為 Auto Scaling 群組的目前容量百分比或容量單位。通常，除非您使用執行個體權重功能，否則容量單位代表一個執行個體。

Amazon EC2 Auto Scaling 支援以下擴展和簡易擴展的調整類型：

- **ChangeInCapacity**：依照指定的數值，增加或減少群組目前的容量。正值即增加容量，負調整值則會減少容量。例如：如果群組目前的容量為 3 且調整值為 5，執行此政策時，會將容量增加 5 個容量單位，總共 8 個容量單位。
- **ExactCapacity**：將群組的目前容量變更為指定的數值。此調整類型應指定非負值。例如：如果群組目前的容量為 3 且調整值為 5，執行此政策時，會將容量變更為 5 個容量單位。

- **PercentChangeInCapacity**：依照指定的百分比，增加或減少群組目前的容量。正值即增加容量，負值則減少容量。例如，如果目前的容量為 10 且調整值為 10%，執行此政策時，會將容量增加 1 個容量單位，總共 11 個容量單位。

Note

若結果值不是整數，會用無條件捨去計算之，如下所示：

- 大於 1 的值無條件捨去取整。例如，12.7 捨入到 12。
- 0 至 1 之間的值捨入到 1。例如，.67 捨入到 1。
- 0 至 -1 之間的值捨入到 -1。例如，-.58 捨入到 -1。
- 小於 -1 的值無條件進位取整。例如，-6.67 捨入到 -6。

透過 **PercentChangeInCapacity**，您也可以使用 **MinAdjustmentMagnitude** 參數來指定要擴展之執行個體的數量下限。例如，假設您建立了一個增加 25% 的政策，並指定最小擴展量為 2 個執行個體。如果您有一個 Auto Scaling 群組具有 4 個執行個體和已執行的擴展政策，4 個執行個體的 25% 即 1 個執行個體。不過，由於您指定最小擴展量為 2，所以會增加 2 執行個體。

當您使用 **執行個體權重** 時，將 **MinAdjustmentMagnitude** 參數設定為非零值的效果會變更。此值以容量單位表示。若要設定要擴展的執行個體數量下限，請將此參數設為至少與最大執行個體權重一樣大的值。

如果您使用執行個體權重，請記住 Auto Scaling 群組的目前容量可能會視需要超過所需的容量。如果要減少的絕對數字，或要減少的百分比數量小於目前容量與所需容量之間的差異，則不會執行任何擴展動作。如果檢視超出警示閾值時擴展政策的結果，您必須將這些行為列入考量。例如，假設所需的容量為 30，而目前的容量為 32。超出警示閾值時，如果擴展政策將所需的容量減少 1，則不會採取任何擴展動作。

執行個體暖機期

對於步進擴展，您可以選擇性指定新啟動的執行個體暖機所需的秒數。在指定的暖機時間結束之前，執行個體不會計入 Auto Scaling 群組的彙總 EC2 執行個體指標。

當執行個體處於暖機期間時，只有在未暖機的執行個體指標值大於政策的警示高閾值時，擴展政策才會向外擴展。

如果群組再次水平擴展，則仍在暖機的執行個體將計為下次水平擴展活動所需的容量一部分。因此，多個警示違規落於相同步進調整範圍，促使單一擴展活動。這種做法的目的是連續的向外擴展 (但並非過度)。

舉例來說，假設您使用兩個步驟建立政策。第一步，當指標值達到 60 時，增加 10%；第二步，當指標值達到 70% 時，增加 30%。Auto Scaling 群組的所需容量和目前容量均為 10。當彙總指標值小於 60 時，所需容量和目前容量保持不變。假設指標值達到 60，因此新增 1 個執行個體 (10 個執行個體的 10%)。接著指標值達到 62，同時新的執行個體仍處於暖機狀態。擴展政策會根據目前容量 (仍為 10) 來計算新的所需容量。不過，群組的所需容量已經增加至 11 個執行個體，所以擴展政策不會進一步增加所需容量。如果指標值為 70，同時新執行個體仍處於預熱狀態，則我們應該新增 3 個執行個體 (10 個執行個體的 30%)。不過，群組所需的容量已經為 11，所以我們只新增 2 個執行個體，適用於新所需的容量為 13 個執行個體。

在進行水平擴展活動時，透過擴展政策啟動的所有縮減活動都會遭到封鎖，直到執行個體完成暖機。當執行個體完成暖機時，如果發生縮減事件，那麼在計算新的所需容量時，目前正在終止過程中的任何執行個體都會計入群組的目前容量。所以，我們不會從 Auto Scaling 群組移除超過必要數量的執行個體。舉例來說，執行個體正在終止時，如果發生警示時超出將所需容量減少 1 的相同步進調整範圍，則不會採取任何擴展動作。

預設值

如果未設定任何值，擴展政策將使用預設值，這是為群組定義之[預設執行個體暖機](#)期的值。如果預設執行個體暖機期為 null，則會回到[預設冷卻時間](#)的值。

考量事項

使用步進和簡易擴展政策時，下列考量適用：

- 考慮您是否能夠準確地預測應用程式上的步進調整，以便使用步進擴展。如果您的擴展指標可按比例提高或降低可擴展目標容量，我們建議您改用目標追蹤擴展政策。您仍然可以選擇使用步進擴展作為其他政策，以進行更進階的設定。例如，您可以設定使用率達到特定層級時更積極的回應。
- 請務必在橫向擴展閾值和縮減閾值之間選擇足夠的邊際，以防止震盪。振盪不穩是指向內縮減和水平擴展無限循環的現象。也就是說，如果採取擴展動作，指標值將會改變，並反向展開另一次擴展動作。

建立橫向擴展的步驟擴展政策

若要為 Auto Scaling 群組建立橫向擴展的步驟擴展政策，請使用下列其中一種方法：

Console

步驟 1：建立指標高閾值的 CloudWatch 警示

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。

2. 如有必要請變更 區域。請在導覽列中選擇 Auto Scaling 群組所在的區域。
3. 在導覽窗格中，選擇 Alarms, All alarms (警示，所有警示)，然後選擇 Create alarm (建立警示)。
4. 選擇 Select metric (選取指標)。
5. 在 All metrics (所有指標) 索引標籤上，選擇 EC2、By Auto Scaling Group (依據 Auto Scaling 群組)，然後在搜尋欄位中輸入 Auto Scaling 群組的名稱。然後，選取 CPUUtilization 並選擇 Select metric (選取指標)。Specify metric and conditions (指定指標和條件) 頁面隨即出現，顯示指標的圖表及其他資訊。
6. 針對 Period (期間)，選擇警示的評估期間，例如 1 分鐘。評估警示時，每個期間都會彙整為一個資料點。

 Note

期間越短會建立更敏感的警示。

7. 在 Conditions (條件) 下，執行下列動作：
 - 對於 Threshold type (閾值類型)，選擇 Static (靜態)。
 - 對於每當CPUUtilization為 時，指定您希望指標的值大於或等於閾值以違反警示。然後，在 than (比) 下，輸入您要設定為超標警示的閾值。
8. 在 Additional configuration (其他設定) 下，請執行下列動作：
 - 針對 Datapoints to alarm (要警示的資料點)，輸入資料點 (評估期間)，在此期間指標值必須符合警示的閾值條件。例如，連續兩個 5 分鐘即表示需時 10 分鐘才會呼叫警示狀態。
 - 對於 Missing data treatment (遺失資料處理)，選擇 Treat missing data as bad (breaching threshold) (將遺失資料視為不良 (違反閾值))。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[設定 CloudWatch 警示如何處理遺失資料](#)。
9. 選擇下一步。

Configure actions (設定動作) 頁面隨即顯示。

10. 在 Notification (通知) 下，選取 Amazon SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

若要讓警示不傳送通知，請選擇 Remove (移動)。

11. 您可以將 Configure actions (設定動作) 頁面的其他區段保留空白。將其他區段保留空白會建立警示，而不會將其與擴展政策建立關聯。然後，您可以從 Amazon EC2 Auto Scaling 主控台將此警示與擴展政策建立關聯。
12. 選擇下一步。
13. 輸入名稱 (例如 Step-Scaling-AlarmHigh-AddCapacity)，選擇性地輸入警示描述，然後選擇 Next (下一步)。
14. 選擇 Create alarm (建立警示)。

使用下列程序，在建立 CloudWatch 警示後從您離開的地方繼續。

步驟 2：建立橫向擴展的步驟擴展政策

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 確認已將擴展限制設定妥當。例如，如果群組已達到所需容量上限，則需要指定新的上限，以便進行橫向擴展。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
4. 在 Automatic scaling (自動擴展) 索引標籤的 Dynamic scaling policies (動態擴展政策) 中，選擇 Create dynamic scaling policy (建立動態擴展政策)。
5. 針對政策類型，選擇步驟擴展，然後指定政策的名稱。
6. 如果是 CloudWatch alarm (CloudWatch 警示)，請選擇您的警示。如果您尚未建立警示，請選擇建立 CloudWatch 警示，並完成先前程序中的步驟 4 到步驟 14 來建立警示。
7. 使用 Take the action (採取動作) 指定此政策將進行的目前群組大小變更。您可以新增特定數量的執行個體，或現有群組大小的百分比，或將群組設為確切大小。

例如，若要建立將群組容量增加 30% 的向外擴展政策，請選擇 Add，30 在下一個欄位中輸入，然後選擇 percent of group。根據預設，此步驟調整的下限即為警示閾值，而上限為無限大正數 (+) 值。

8. 若要新增另一個步進，請選擇 Add step (新增步進)，然後定義擴展量，以及相對於警示閾值的步進下限與上限。
9. 若要設定擴展的執行個體數量下限，請更新 Add capacity units in increments of at least 1 capacity units (以至少 1 個容量單位的增量幅度來新增容量單位) 中的數值欄位。
10. (選用) 對於執行個體暖機期，視需要更新執行個體暖機期值。

11. 選擇建立。

AWS CLI

若要建立橫向擴展（增加容量）的步驟擴展政策，您可以使用下列範例命令。將每個#####替換為自己的資訊。

當您使用時 AWS CLI，會先建立步進擴展政策，向 Amazon EC2 Auto Scaling 提供指示，說明如何在指標值增加時向外擴展。然後，您可以透過識別要監看的指標、定義警示的指標高閾值和其他詳細資訊，以及將警示與擴展政策建立關聯來建立警示。

步驟 1：建立向外擴展的政策

使用以下 [put-scaling-policy](#) 命令，建立名為 my-step-scale-out-policy 的步進擴展政策，其調整類型為 PercentChangeInCapacity，可依據以下步進調整增加群組容量 (假設 CloudWatch 警示閾值為 60%)：

- 當指標值大於或等於 60% 但小於 75% 時，將執行個體數增加 10%
- 當指標值大於或等於 75% 但小於 85% 時，將執行個體數增加 20%
- 當指標值大於或等於 85% 時，將執行個體數增加 30%

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
MetricIntervalLowerBound=0.0,MetricIntervalUpperBound=15.0,ScalingAdjustment=10 \
MetricIntervalLowerBound=15.0,MetricIntervalUpperBound=25.0,ScalingAdjustment=20 \
MetricIntervalLowerBound=25.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

記錄政策的 Amazon Resource Name (ARN)。您需要其來建立政策的 CloudWatch 警示。

```
{
  "PolicyARN":
    "arn:aws:autoscaling:region:123456789012:scalingPolicy:4ee9e543-86b5-4121-b53b-aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-step-scale-in-policy"
```



```
}
```

步驟 2：為指標高閾值建立 CloudWatch 警示

使用以下 CloudWatch [put-metric-alarm](#) 命令建立警示，依據 CPU 平均閾值在至少兩個連續的兩分鐘評估期間內達到 60% 時，增加 Auto Scaling 群組的大小。若要使用自訂指標，請於 `--metric-name` 中指定它的名稱，於 `--namespace` 中指定它的命名空間。

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-AddCapacity \  
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \  
  --period 120 --evaluation-periods 2 --threshold 60 \  
  --comparison-operator GreaterThanOrEqualToThreshold \  
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \  
  --alarm-actions PolicyARN
```

建立向內擴展的步驟擴展政策

若要為 Auto Scaling 群組建立縮減的步驟擴展政策，請使用下列其中一種方法：

Console

步驟 1：為指標低閾值建立 CloudWatch 警示

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 如有必要請變更 區域。請在導覽列中選擇 Auto Scaling 群組所在的區域。
3. 在導覽窗格中，選擇 Alarms, All alarms (警示，所有警示)，然後選擇 Create alarm (建立警示)。
4. 選擇 Select metric (選取指標)。
5. 在 All metrics (所有指標) 索引標籤上，選擇 EC2、By Auto Scaling Group (依據 Auto Scaling 群組)，然後在搜尋欄位中輸入 Auto Scaling 群組的名稱。然後，選取 CPUUtilization 並選擇 Select metric (選取指標)。Specify metric and conditions (指定指標和條件) 頁面隨即出現，顯示指標的圖表及其他資訊。
6. 針對 Period (期間)，選擇警示的評估期間，例如 1 分鐘。評估警示時，每個期間都會彙整為一個資料點。

 Note

期間越短會建立更敏感的警示。

7. 在 Conditions (條件) 下，執行下列動作：

- 對於 Threshold type (閾值類型)，選擇 Static (靜態)。
- 對於每當 **CPUUtilization** 為 時，指定您希望指標的值小於或等於閾值，以違反警示。然後，在 than (比) 下，輸入您要設定為超標警示的閾值。

 Important

若要讓警示與縮減政策（指標低）搭配使用，請確定您未選擇大於或等於閾值。

8. 在 Additional configuration (其他設定) 下，請執行下列動作：

- 針對 Datapoints to alarm (要警示的資料點)，輸入資料點 (評估期間)，在此期間指標值必須符合警示的閾值條件。例如，連續兩個 5 分鐘即表示需時 10 分鐘才會呼叫警示狀態。
- 對於 Missing data treatment (遺失資料處理)，選擇 Treat missing data as bad (breaching threshold) (將遺失資料視為不良 (違反閾值))。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[設定 CloudWatch 警示如何處理遺失資料](#)。

9. 選擇下一步。

Configure actions (設定動作) 頁面隨即顯示。

10. 在 Notification (通知) 下，選取 Amazon SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

若要讓警示不傳送通知，請選擇 Remove (移動)。

11. 您可以將 Configure actions (設定動作) 頁面的其他區段保留空白。將其他區段保留空白會建立警示，而不會將其與擴展政策建立關聯。然後，您可以從 Amazon EC2 Auto Scaling 主控台將此警示與擴展政策建立關聯。

12. 選擇下一步。

13. 輸入名稱 (例如 Step-Scaling-AlarmLow-RemoveCapacity)，選擇性地輸入警示描述，然後選擇 Next (下一步)。

14. 選擇 Create alarm (建立警示)。

使用下列程序在建立 CloudWatch 警示後，從您離開的地方繼續。

步驟 2：建立縮減的步驟擴展政策

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 確認已將擴展限制設定妥當。例如，如果您群組的所需容量已達最低容量，則需要指定新的最低容量才能縮減。如需詳細資訊，請參閱[設定 Auto Scaling 群組的擴展限制](#)。
4. 在 Automatic scaling (自動擴展) 索引標籤的 Dynamic scaling policies (動態擴展政策) 中，選擇 Create dynamic scaling policy (建立動態擴展政策)。
5. 針對政策類型，選擇步驟擴展，然後指定政策的名稱。
6. 如果是 CloudWatch alarm (CloudWatch 警示)，請選擇您的警示。如果您尚未建立警示，請選擇建立 CloudWatch 警示，並完成先前程序中的步驟 4 到步驟 14 來建立警示。
7. 使用 Take the action (採取動作) 指定此政策將進行的目前群組大小變更。您可以移除特定數量的執行個體，或現有群組大小的百分比，或將群組設為確切大小。

例如，若要建立縮減政策以將群組容量減少兩個執行個體，請選擇 Remove，2 在下一個欄位中輸入，然後選擇 capacity units。根據預設，此步進調整的上限即為警示閾值，而下限為無限小負數 (-) 值。

8. 若要新增另一個步進，請選擇 Add step (新增步進)，然後定義擴展量，以及相對於警示閾值的步進下限與上限。
9. 選擇建立。

AWS CLI

若要建立縮減（減少容量）的步驟擴展政策，您可以使用下列範例命令。將每個#####替換為自己的資訊。

當您使用時 AWS CLI，會先建立步進擴展政策，向 Amazon EC2 Auto Scaling 提供指示，說明如何在指標值減少時縮減規模。然後，您可以透過識別要監看的指標、定義警示的指標低閾值和其他詳細資訊，以及將警示與擴展政策建立關聯，來建立警示。

步驟 1：建立縮減的政策

使用以下 [put-scaling-policy](#) 命令，建立名為 my-step-scale-in-policy 的步進擴展政策，其調整類型為 ChangeInCapacity，會在關聯的 CloudWatch 警示超出指標低閾值時將群組的容量減少 2 個執行個體。

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

記錄政策的 Amazon Resource Name (ARN)。您需要其來建立政策的 CloudWatch 警示。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-step-scale-out-policy"
}
```

步驟 2：建立指標低閾值的 CloudWatch 警示

使用以下 CloudWatch [put-metric-alarm](#) 命令建立警示，依據 CPU 平均閾值在至少兩個連續的兩分鐘評估期間內達到 40% 時，減少 Auto Scaling 群組的大小。若要使用自訂指標，請於 --metric-name 中指定它的名稱，於 --namespace 中指定它的命名空間。

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmLow-RemoveCapacity \
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \
  --period 120 --evaluation-periods 2 --threshold 40 \
  --comparison-operator LessThanOrEqualToThreshold \
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \
  --alarm-actions PolicyARN
```

簡易擴展政策

下列範例示範如何使用 CLI 命令來建立簡單的擴展政策。它們會保留在本文件中，做為任何想要使用它們之客戶的參考，但我們建議您改用目標追蹤或步進擴展政策。

與步進擴展政策類似，簡易擴展政策會要求您為擴展政策建立 CloudWatch 警示。在您建立的政策中，您還必須定義是否要新增或移除執行個體，以及要將群組設定為確切大小。

步驟擴展政策和簡單擴展政策之間的主要差異之一，是您在步驟擴展政策中取得的步驟調整。透過步進擴展，您可以根據您指定的步進調整，對群組的大小進行更大或更小的變更。

簡單的擴展政策也必須等待進行中的擴展活動或運作狀態檢查取代完成，以及[冷卻時間](#)結束，才能回應其他警示。相反地，使用步進擴展時，即使正在進行擴展活動或運作狀態檢查取代，政策仍會繼續回應其他警示。這表示 Amazon EC2 Auto Scaling 會在收到警示訊息時評估所有警示違規。因此，我們建議您改用步進擴展政策，即使您只有單一擴展調整。

Amazon EC2 Auto Scaling 最初支援的只有簡易擴展政策。如果您在引入目標追蹤和步驟擴展政策之前建立了擴展政策，您的政策會被視為簡單的擴展政策。

為向外擴展建立簡單的擴展政策

使用以下 [put-scaling-policy](#) 命令建立名為 my-simple-scale-out-policy 的簡易擴展政策，其調整類型為 PercentChangeInCapacity，會在關聯的 CloudWatch 警示超出指標高閾值時將群組容量增加 30%。

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \  
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \  
  --adjustment-type PercentChangeInCapacity
```

記錄政策的 Amazon Resource Name (ARN)。您需要其來建立政策的 CloudWatch 警示。

為縮減建立簡單的擴展政策

使用以下 [put-scaling-policy](#) 命令建立名為 my-simple-scale-in-policy 的簡易擴展政策，其調整類型為 ChangeInCapacity，會在關聯的 CloudWatch 警示超出指標低閾值時將群組容量減少一個執行個體。

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \  
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \  
  --adjustment-type ChangeInCapacity --cooldown 180
```

記錄政策的 Amazon Resource Name (ARN)。您需要其來建立政策的 CloudWatch 警示。

擴展 Amazon EC2 Auto Scaling 冷卻時間

Important

依最佳實務，建議您不要使用簡易擴展政策和擴展冷卻時間。目標追蹤擴展政策或步進擴展政策的擴展效能更佳。針對隨擴展指標減少或增加而依比例變更您 Auto Scaling 群組大小的擴展政策，我們建議使用[目標追蹤](#)而非簡易擴展或步進擴展。

當您為 Auto Scaling 群組建立簡易擴展政策時，我們建議您同時設定擴展冷卻時間。

在 Auto Scaling 群組啟動或終止執行個體後，其會先等待冷卻時間結束，然後才能啟動簡易擴展政策啟動的任何進一步擴展活動。冷卻時間的目的是讓您的 Auto Scaling 群組穩定，並防止它在先前擴展活動的效果顯示之前啟動或終止其他執行個體。

例如，假設 CPU 使用率的簡易擴展政策建議啟動兩個執行個體。Amazon EC2 Auto Scaling 會啟動兩個執行個體，然後暫停擴展活動，直到冷卻時間結束。在冷卻時間結束後，任何簡易擴展政策啟動的擴展活動都可以繼續。如果 CPU 使用率再次違反高警示閾值，Auto Scaling 群組會再次水平擴展，且冷卻時間也會再次作用。但是，如果兩個執行個體已足夠降低指標值，群組便會維持目前的大小。

目錄

- [考量事項](#)
- [生命週期關聯可能會導致額外延遲](#)
- [變更預設冷卻時間](#)
- [為特定的簡單擴展政策設定冷卻時間](#)

考量事項

下列考量適用於使用簡易擴展政策和擴展冷卻：

- 目標追蹤和步進擴展政策可以立即啟動水平擴展活動，而無需等待冷卻時間結束。反之，每當 Auto Scaling 群組啟動執行個體時，個別執行個體都會有暖機期。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。
- 當橫向擴展活動正在進行時，會封鎖所有目標追蹤和步進擴展向內擴展活動，直到執行個體完成暖機。當 Auto Scaling 群組處於冷卻時間時，縮減活動也可能延遲。
- 當已排定的動作在已排定的時間開始時，它還會立即啟動擴展活動，而無須等候冷卻時間結束。

- 如果執行個體狀態不佳，Amazon EC2 Auto Scaling 不會等待冷卻時間結束，然後再取代此運作狀態不佳的執行個體。
- 當啟動或終止多個執行個體時，冷卻時間 (預設冷卻時間或擴展政策特定冷卻時間) 會在最後一個執行個體完成啟動或終止時，開始生效。
- 在手動擴展 Auto Scaling 群組時，預設並不會等待冷卻時間完成。不過，當您使用 AWS CLI 或 SDK 手動擴展時，您可以覆寫此行為並遵循預設冷卻時間。
- 依預設，Elastic Load Balancing 會等待 300 秒以完成取消註冊 (連接耗盡) 程序。如果該群組在 Elastic Load Balancing 負載平衡器之後，其會等待終止執行個體取消註冊，再開始冷卻時間。

生命週期關聯可能會導致額外延遲

如果叫用 [lifecycle hook](#)，冷卻時間會在您完成生命週期動作後或逾時期間結束後開始。例如，假設有一個 Auto Scaling 群組，其具備可用於執行個體啟動的 lifecycle hook 的。當應用程式提高需求，群組啟動執行個體以增加容量。由於具備一個 lifecycle hook，執行個體進入等待狀態，且擴展活動會由於簡易擴展政策而暫停。當執行個體進入 InService 狀態時，冷卻時間開始。在冷卻時間結束後，會繼續簡易擴展政策活動。

啟用 Elastic Load Balancing 時，為了縮減規模，當為終止選取的執行個體開始連線耗盡 (取消註冊延遲) 時，冷卻時間就會開始。冷卻時間不會等待連線耗盡完成，也不會等待生命週期掛鉤完成其動作。這表示只要縮減事件的結果反映在群組的容量中，任何因簡單擴展政策而導致的擴展活動都可以繼續。否則，等待完成所有三個活動 (連接耗盡、lifecycle hook 和冷卻時間) 會大幅增加 Auto Scaling 群組暫停擴展所需的時間。

變更預設冷卻時間

一開始在 Amazon EC2 Auto Scaling 主控台中建立 Auto Scaling 群組時，您無法設定預設冷卻時間。依預設，此冷卻時間期間設為 300 秒 (5 分鐘)。如有需要，您可以在建立群組後進行更新。

變更預設冷卻時間 (主控台)

建立 Auto Scaling 群組後，在 Details (詳細資訊) 索引標籤上，依次選取 Advanced configurations (進階組態)、Edit (編輯)。對於 Default cooldown (預設冷卻時間)，根據您的執行個體啟動時間或其他應用程式需求來選擇時間量。

變更預設冷卻時間 (AWS CLI)

使用以下命令變更新的或現有 Auto Scaling 群組的預設冷卻時間。如果未定義預設冷卻時間，則使用預設值 300 秒。

- [create-auto-scaling-group](#)
- [update-auto-scaling-group](#)

要確認預設冷卻時間的值，請使用 [describe-auto-scaling-groups](#) 命令。

為特定的簡單擴展政策設定冷卻時間

依預設，所有簡易擴展政策都會使用為 Auto Scaling 群組定義的預設冷卻時間。如要設定特定簡易擴展政策的冷卻時間，請在建立或更新政策時使用選用的冷卻參數。在為政策指定冷卻時間時，它會覆寫預設冷卻時間。

擴展政策特定冷卻時間的常見用途之一是擴展政策。由於該政策終止執行個體，Amazon EC2 Auto Scaling 需要更少的時間來確定是否要終止額外的執行個體。終止執行個體的操作應會比啟動執行個體更快。因此，預設的 300 秒冷卻時間過長。在這種情況下，擴展政策特定的冷卻時間，其縮減政策的值較低，可以透過允許群組更快地縮減規模，協助您降低成本。

要在主控台中建立或更新簡易擴展政策，請在建立群組後選擇 Automatic scaling (自動擴展) 索引標籤。若要使用 建立或更新簡單的擴展政策 AWS CLI，請使用 [put-scaling-policy](#) 命令。如需詳細資訊，請參閱 [步進和簡易擴展政策](#)。

以 Amazon SQS 為基礎的擴展政策

Important

下列資訊和步驟說明在將佇列屬性作為自訂指標發佈到 CloudWatch 之前，如何使用 ApproximateNumberOfMessages 佇列屬性計算每個執行個體的 Amazon SQS 佇列待辦項目。但是，您現在可以使用指標數學來節省發佈自己指標的成本和精力。如需詳細資訊，請參閱 [使用指標數學建立目標追蹤擴展政策](#)。

您可以擴展 Auto Scaling 群組，以回應 Amazon Simple Queue Service (Amazon SQS) 佇列中系統負載的變更。若要進一步了解如何使用 Amazon SQS，請參閱 [《Amazon Simple Queue Service 開發人員指南》](#)。

在某些情境下，您可能會考慮進行擴展，以回應 Amazon SQS 佇列中的活動。例如，假設您有一個 web 應用程式，可讓使用者上傳影像及在線上使用這些影像。在此情境下，每個影像都需要調整大小和編碼，才能進行發佈。應用程式會在 Auto Scaling 群組內的 EC2 執行個體上執行，並且會設為處理您的典型上傳速率。運作狀態不佳的執行個體會中止且取代，為維持目前執行個體數量。應用程式會將

影像的新點陣圖資料置放在 SQS 佇列中以進行處理。它會處理影像，然後發佈處理後的影像，供使用者瀏覽。如果要上傳的影像數量不會隨時間變化，此案例的架構便能夠良好的運作。但如果上傳的數量會隨時間變更，建議您考慮使用動態擴展來擴展您 Auto Scaling 群組的容量。

目錄

- [搭配正確的指標使用目標追蹤](#)
- [限制](#)
- [根據 Amazon SQS 設定擴展](#)
- [Amazon SQS 和執行個體縮減保護](#)

搭配正確的指標使用目標追蹤

如果您使用以自訂 Amazon SQS 佇列指標為基礎的目標追蹤擴展政策，動態擴展可以更有效地調整至您應用程式的需求曲線。如需為目標追蹤選擇指標的詳細資訊，請參閱 [選擇 Metrics \(指標\)](#)。

使用如 `ApproximateNumberOfMessagesVisible` 等 CloudWatch Amazon SQS 指標進行目標追蹤的問題在於，佇列中的訊息數不能根據處理佇列訊息的 Auto Scaling 群組大小依比例進行變更。這是因為您 SQS 佇列中的訊息數不只會定義所需要的執行個體數。Auto Scaling 群組中的執行個體數會受到多個因素影響，包括其處理訊息所花費的時間，以及可接受的延遲量 (佇列延遲)。

為解決這些狀況，可使用帶有目標值的 backlog per instance (每個執行個體的待處理項目) 指標，作為要維持的 acceptable backlog per instance (每個執行個體的可接受待處理項目)。您可以透過以下列方式計算這些數字：

- 每個執行個體的待處理項目：如要計算每個執行個體的待處理項目，請從 `ApproximateNumberOfMessages` 佇列屬性開始，以判斷 SQS 佇列的長度 (可從佇列供擷取使用的訊息數)。將此數量除以機群的執行容量 (對 Auto Scaling 群組而言是處於 `InService` 狀態的執行個體數量)，求出每個執行個體的待處理項目。
- 每個執行個體的可接受待處理項目：如要計算您的目標值，請先判斷您應用程式針對延遲可接受的應用程式。然後，將可接受延遲值，除以 EC2 執行個體處理訊息所需的平均時間。

例如，假設您目前有一個 Auto Scaling 群組，其中包含 10 個執行個體，且佇列 (`ApproximateNumberOfMessages`) 中的可見訊息數量為 1500。如果每個訊息的平均處理時間為 0.1 秒，最長可接受的延遲為 10 秒，則每個執行個體的可接受待處理項目 $10/0.1$ ，等於 100 個訊息。這表示 100 是目標追蹤政策的目標值。當每個執行個體的待處理項目達到目標值時，就會發生橫向擴展事件。因為每個執行個體的待處理項目數量已經為 150 個訊息 (1500 個訊息/10 個執行個體)，您的群組便會橫向擴展，並橫向擴展五個執行個體來維持與目標值的比例。

下列程序示範如何發佈自訂指標及建立目標追蹤擴展政策，設定您的 Auto Scaling 群組來根據這些計算進行擴展。

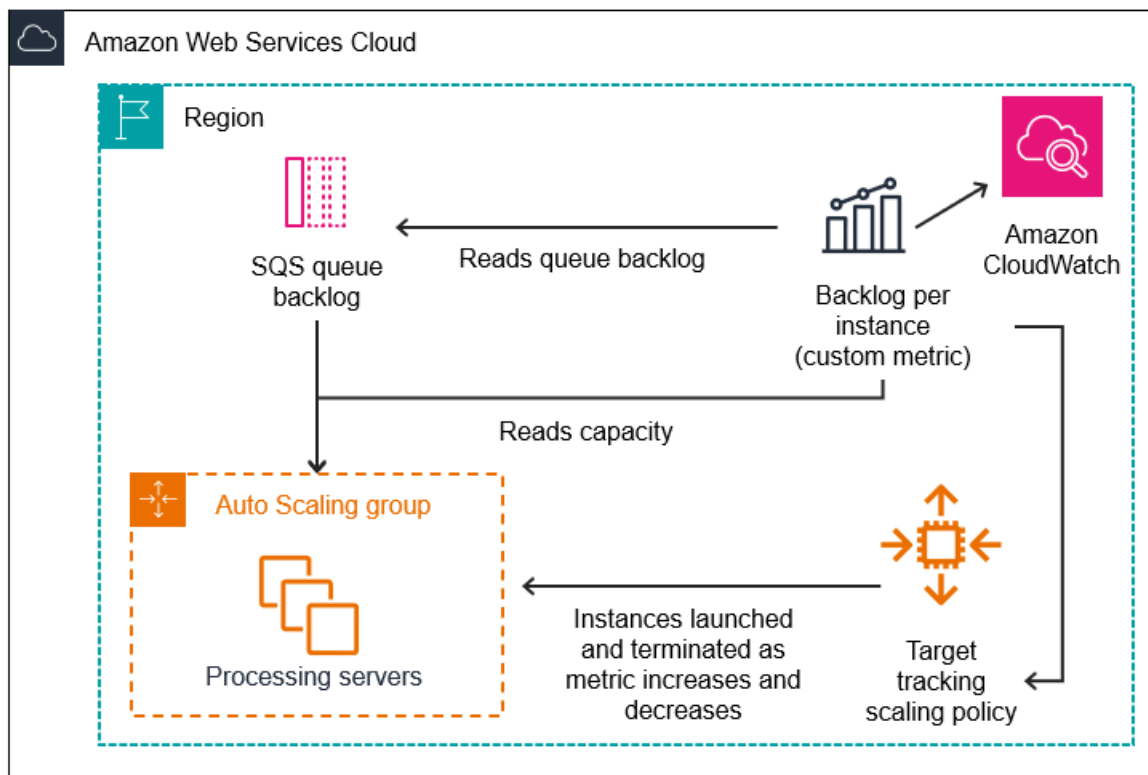
⚠ Important

請記住，若要降低成本，請改用指標數學。如需詳細資訊，請參閱[使用指標數學建立目標追蹤擴展政策](#)。

這組態有三個重要部分：

- 用來管理 EC2 執行個體的 Auto Scaling 群組，主要目的在處理來自 SQS 佇列中的訊息。
- 傳送給 Amazon CloudWatch 的自訂指標，用來 Auto Scaling 群組每個 EC2 執行個體佇列中的訊息數量。
- 目標追蹤擴展政策設定 Auto Scaling 群組會根據自訂指標和設定的目標值來進行擴展。CloudWatch 警示呼叫擴展政策。

下圖說明此組態的架構。



限制

您必須使用 AWS CLI 或 SDK 將自訂指標發佈至 CloudWatch。然後，您可以使用 監控指標 AWS 管理主控台。

在下列各節中，您會 AWS CLI 針對需要執行的任務使用。例如，若要取得反映佇列目前使用情況的指標資料，您可以使用 SQS [get-queue-attributes](#) 命令。

根據 Amazon SQS 設定擴展

下列程序說明如何根據 Amazon SQS 設定自動擴展。您將了解如何建立 CloudWatch 自訂指標、如何使用 設定目標追蹤政策 AWS CLI，以及如何測試您的組態。

開始之前，請確定您已 AWS CLI [安裝並設定](#)。此外，您必須擁有要使用的 Amazon SQS 佇列。下列任務假設您已有佇列（標準或 FIFO）、Auto Scaling 群組，以及執行使用佇列之應用程式的 EC2 執行個體。

如需有關 Amazon SQS 主題的詳細資訊，請參閱 [《Amazon Simple Queue Service 開發人員指南》](#)。

任務

- [步驟 1：建立 CloudWatch 自訂指標](#)
- [步驟 2：建立目標追蹤擴展政策](#)
- [步驟 3：測試您的擴展政策](#)

步驟 1：建立 CloudWatch 自訂指標

自訂指標是您定義使用指標名稱和命名空間定義的。自訂指標命名空間無法以 AWS/ 為開頭。如需更多發佈自訂指標的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [發佈自訂指標](#) 主題。

遵循此程序，首先從 AWS 您的帳戶讀取資訊來建立自訂指標。然後，如先前小節所建議，計算每個執行個體的待處理項目指標。最後，以 1 分鐘粒度將此數值發佈到 CloudWatch。我們強烈建議在可能的情況下使用 1 分鐘粒度擴展指標，確保能夠更快地回應系統負載中的變更。

建立 CloudWatch 自訂指標 (AWS CLI)

1. 使用 SQS [get-queue-attributes](#) 命令，取得佇列中等待的訊息數量 (ApproximateNumberOfMessages)。

```
aws sqs get-queue-attributes --queue-url https://  
sqs.region.amazonaws.com/123456789/MyQueue \
```

```
--attribute-names ApproximateNumberOfMessages
```

2. 使用 [describe-auto-scaling-group](#) 命令，取得群組的執行容量，此數值是處於 InService 生命週期狀態的執行個體數量。此命令傳回的 Auto Scaling 群組的執行個體及其生命週期狀態。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

3. 透過將可從群組擷取的大約訊息數除以機群的執行容量，來計算每個執行個體的待處理項目。
4. 建立每分鐘執行一次的指令碼，以擷取每個執行個體值的待處理項目，並將其發佈到 CloudWatch 自訂指標。發佈 Auto Scaling 時，需要指定指標的名稱、命名空間、單位、值以及零個或多個維度。維度由維度名稱和維度值組成。

若要發佈自訂指標，請用偏好的指標名稱、指標值、命名空間 (只要不是以「AWS」開頭即可) 和維度 (選擇性) 取代以 `##` 顯示的預留位置值，然後執行下列 [put-metric-data](#) 命令。

```
aws cloudwatch put-metric-data --metric-name MyBacklogPerInstance --  
namespace MyNamespace \  
  --unit None --value 20 --  
dimensions MyOptionalMetricDimensionName=MyOptionalMetricDimensionValue
```

在您的應用程式發送所需的指標後，資料會傳送到 CloudWatch。CloudWatch 主控台中會顯示指標。您可以登入 AWS 管理主控台 並導覽至 CloudWatch 頁面來存取它。然後，導覽至指標頁面或是使用搜尋方塊進行搜尋以檢視指標。如需檢視指標的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[檢視可用指標](#)。

步驟 2：建立目標追蹤擴展政策

您建立的指標現在可以新增至目標追蹤擴展政策。

建立目標追蹤擴展政策 (AWS CLI)

1. 針對主目錄中名為 `config.json` 的 JSON 檔案的擴展政策和自訂指標規範，使用以下 `cat` 命令來儲存其目標值。將每個 `#####` 替換為自己的資訊。對於 `TargetValue`，計算每個執行個體的可接受待處理項目指標，然後在這裡輸入。要計算這個數值，需要決定一般延遲值，然後以處理訊息所花的平均時間將其除之，如之前章節所述。

如果您沒有為在步驟 1 中建立的指標指定任何維度，請勿在自訂指標規範中包含任何維度。

```
$ cat ~/config.json  
{
```

```

    "TargetValue":100,
    "CustomizedMetricSpecification":{
      "MetricName":"MyBacklogPerInstance",
      "Namespace":"MyNamespace",
      "Dimensions":[
        {
          "Name":"MyOptionalMetricDimensionName",
          "Value":"MyOptionalMetricDimensionValue"
        }
      ],
      "Statistic":"Average",
      "Unit":"None"
    }
  }
}

```

2. 使用 [put-scaling-policy](#) 命令與先前步驟中建立的 config.json 檔案來建立您的擴展政策。

```

aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://~/config.json

```

這會建立兩個警示：一個用於擴增，而另一個用於縮減。這也會傳回向 CloudWatch 註冊的政策 Amazon Resource Name (ARN)，每當超出指標閾值時，CloudWatch 即會使用 ARN 來呼叫擴展。

步驟 3：測試您的擴展政策

設定完成後，確認您的擴展政策可正常運作。您可以利用提高 SQS 佇列中的訊息數量來測試該政策，然後驗證 Auto Scaling 群組是否有啟動一個額外的 EC2 執行個體。您也可以利用減低 SQS 佇列中的訊息數量來測試縮減政策，然後驗證 Auto Scaling 群組是否有終止一個 EC2 執行個體。

測試擴增函數

1. 請遵循[建立 Amazon SQS 標準佇列、傳送訊息](#)或[建立 Amazon SQS FIFO 佇列，以及傳送訊息](#)以將訊息新增至佇列中的步驟。請確定您已調升佇列中的訊息數量，因此每個執行個體的待處理項目指標才會達到目標值。

這變更可能需要幾分鐘的時間來呼叫警示。

2. 使用 [describe-auto-scaling-group](#) 命令，驗證群組是否已啟動執行個體。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

測試縮減函數

1. 依照[接收和刪除訊息（主控台）](#)中的步驟，從佇列刪除訊息。請確定您已調降佇列中的訊息數量，因此每個執行個體的待處理項目指標才會低於目標值。

這變更可能需要幾分鐘的時間來呼叫警示。

2. 使用 [describe-auto-scaling-group](#) 命令，驗證群組是否已終止執行個體。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Amazon SQS 和執行個體縮減保護

在執行個體終止時尚未處理的訊息會傳回 SQS 佇列，由仍在執行的執行個體處理。對於執行長時間執行任務的應用程式，您可以選擇性地使用執行個體縮減保護，控制 Auto Scaling 群組縮減時要終止哪些佇列工作者。

下列虛擬程式碼顯示可保護長時間執行、佇列導向的工作者處理程序免於縮減終止的一種方法。

```
while (true)
{
    SetInstanceProtection(False);
    Work = GetNextWorkUnit();
    SetInstanceProtection(True);
    ProcessWorkUnit(Work);
    SetInstanceProtection(False);
}
```

如需詳細資訊，請參閱[設計您的應用程式以正常處理執行個體終止](#)。

驗證 Auto Scaling 群組的擴展活動

在 Amazon EC2 主控台的 Amazon EC2 Auto Scaling 區段中，Auto Scaling 群組的 Activity history (活動歷史記錄) 可以讓您檢視進行中的擴展活動的目前狀態。擴展活動完成後，您可以查看其是否成功。建立 Auto Scaling 群組或向現有群組新增擴展條件時，此功能特別實用。

當您向 Auto Scaling 群組新增目標追蹤、步驟或簡單擴展政策時，Amazon EC2 Auto Scaling 會立即開始針對該指標評估政策。當指標超過閾值達到指定的評估期間數，指標警示會進入 ALARM 狀態。這表示擴展政策可能會在建立之後立即產生擴展活動。Amazon EC2 Auto Scaling 調整所需容量以回應擴展政策之後，即可在您的帳戶中驗證擴展活動。如果要接收來自 Amazon EC2 Auto Scaling 的電子郵件通知，通知您有關擴展活動的資訊，則請遵循 [Amazon EC2 Auto Scaling 的 Amazon SNS 通知選項 Amazon EC2 Auto Scaling](#) 中的指示。

Tip

在下列程序中，您可以查看 Auto Scaling 群組的 Activity history (活動歷史記錄) 和 Instances (執行個體) 部分。在這兩個部分中，已顯示了指定的資料欄。要顯示隱藏的資料欄列或變更顯示的資料列，請選擇每個部分右上角的齒輪圖示以開啟偏好設定模式，視需要更新設定，然後選擇 Confirm (確認)。

檢視 Auto Scaling 群組的擴展活動 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所在的區域。
3. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 中，Status (狀態) 欄位會顯示 Auto Scaling 群組是否已成功啟動或終止執行個體，或擴展活動是否仍在進行中。
5. (選用) 如果您有大量擴展活動，則可以選擇活動歷史記錄頂部邊緣的 > 圖示以查看擴展活動的下一頁。
6. 在 Instance management (執行個體管理) 索引標籤 Instances (執行個體) 下的 Lifecycle (生命週期) 欄會包含執行個體的狀態。在執行個體啟動並完成任何 lifecycle hook 後，它的生命週期狀態會變更為 InService。Health status (運作狀態) 欄會顯示對執行個體進行 EC2 執行個體運作狀態檢查的結果。

檢視 Auto Scaling 群組的擴展活動 (AWS CLI)

使用以下 [describe-scaling-activities](#) 命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下為範例輸出。

擴展活動依開始時間排序。首先描述仍在進行中的活動。

```
{
  "Activities": [
    {
      "ActivityId": "5e3a1f47-2309-415c-bfd8-35aa06300799",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-06c4794c2499af1df",
      "Cause": "At 2020-02-11T18:34:10Z a monitor alarm TargetTracking-my-asg-AlarmLow-b9376cab-18a7-4385-920c-dfa3f7783f82 in state ALARM triggered policy my-target-tracking-policy changing the desired capacity from 3 to 2. At 2020-02-11T18:34:31Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 2. At 2020-02-11T18:34:31Z instance i-06c4794c2499af1df was selected for termination.",
      "StartTime": "2020-02-11T18:34:31.268Z",
      "EndTime": "2020-02-11T18:34:53Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a\"...}",
      "AutoScalingGroupARN": "arn"
    },
    ...
  ]
}
```

如需輸出中欄位的說明，請參閱《Amazon EC2 Auto Scaling API 參考》中的[活動](#)。

如需協助擷取已刪除群組的擴展活動，以及可能會遇到的錯誤類型和處理方式的相關資訊，請參閱 [對 Amazon EC2 Auto Scaling 中的問題進行故障診斷](#)。

停用 Auto Scaling 群組的擴展政策

本主題說明如何暫時停用擴展政策，使其不會啟動變更 Auto Scaling 群組包含的執行個體數量。停用擴展政策時，會保留組態詳細資訊，以便您快速重新啟用政策。比起在不需要時暫時刪除政策，然後稍後重新建立，這個方法比較容易。

停用擴展政策時，Auto Scaling 群組不會針對停用擴展政策時違規的指標警示進行擴增或縮減。不過，任何仍在進行中的擴展活動都不會停止。

請注意，停用的擴展政策仍會計入您可以加入 Auto Scaling 群組的擴展政策數量的配額。

停用擴展政策 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Automatic scaling (自動擴展) 索引標籤中，在 Dynamic scaling policies (動態擴展政策) 中，選取所需擴展政策右上角的核取方塊。
4. 捲動至 Dynamic scaling policies (動態擴展政策) 區段，然後選擇 Actions (動作)、Disable (停用)。

當您準備好重新啟用擴展政策時，請重複這些步驟，然後選擇 Actions (動作)、Enable (啟用)。重新啟用擴展政策之後，如果目前有任何警示處於 ALARM 狀態，您的 Auto Scaling 群組可能會立即啟動擴展動作。

停用擴展政策 (AWS CLI)

使用 [put-scaling-policy](#) 命令與 `--no-enabled` 選項，如下所示。指定命令中的所有選項，就像您在建立政策時指定這些選項一樣。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \  
  --policy-name my-scaling-policy --policy-type TargetTrackingScaling \  
  --estimated-instance-warmup 360 \  
  --target-tracking-configuration '{ "TargetValue": 70,  
  "PredefinedMetricSpecification": { "PredefinedMetricType":  
    "ASGAverageCPUUtilization" } }' \  
  --no-enabled
```

重新啟用擴展政策 (AWS CLI)

使用 [put-scaling-policy](#) 命令與 `--enabled` 選項，如下所示。指定命令中的所有選項，就像您在建立政策時指定這些選項一樣。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \  
  --policy-name my-scaling-policy --policy-type TargetTrackingScaling \  
  --estimated-instance-warmup 360 \  
  --target-tracking-configuration '{ "TargetValue": 70,  
  "PredefinedMetricSpecification": { "PredefinedMetricType":  
    "ASGAverageCPUUtilization" } }' \  
  --enabled
```

```
--enabled
```

描述擴展政策 (AWS CLI)

使用 [describe-policies](#) 命令，驗證擴展政策的啟用狀態。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg \  
--policy-names my-scaling-policy
```

以下為範例輸出。

```
{  
  "ScalingPolicies": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "PolicyName": "my-scaling-policy",  
      "PolicyARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scalingPolicy:1d52783a-b03b-4710-  
bb0e-549fd64378cc:autoScalingGroupName/my-asg:policyName/my-scaling-policy",  
      "PolicyType": "TargetTrackingScaling",  
      "StepAdjustments": [],  
      "Alarms": [  
        {  
          "AlarmName": "TargetTracking-my-asg-  
AlarmHigh-9ca53fdd-7cf5-4223-938a-ae1199204502",  
          "AlarmARN": "arn:aws:cloudwatch:us-  
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-  
ae1199204502"  
        },  
        {  
          "AlarmName": "TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-  
abe0-1cf8b9de6d6c",  
          "AlarmARN": "arn:aws:cloudwatch:us-  
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-  
abe0-1cf8b9de6d6c"  
        }  
      ],  
      "TargetTrackingConfiguration": {  
        "PredefinedMetricSpecification": {  
          "PredefinedMetricType": "ASGAverageCPUUtilization"  
        },  
        "TargetValue": 70.0,  
        "DisableScaleIn": false  
      }  
    }  
  ]  
}
```



```
    },  
    "Enabled": true  
  }  
]  
}
```

刪除 Auto Scaling 群組的擴展政策

在您不再需要擴展政策之後，可以將其刪除。根據擴展政策的類型，您可能也需要刪除 CloudWatch 警示。刪除目標追蹤擴展政策也會刪除任何相關聯的 CloudWatch 警示。刪除步進擴展政策或簡易擴展政策會刪除基礎警示動作，但不會刪除 CloudWatch 警示，即使其不再有相關聯的動作，也是如此。

刪除擴展政策 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Automatic scaling (自動擴展) 索引標籤中，在 Dynamic scaling policies (動態擴展政策) 中，選取所需擴展政策右上角的核取方塊。
4. 捲動至 Dynamic scaling policies (動態擴展政策) 區段，然後選擇 Actions (動作)、Delete (刪除)。
5. 出現確認提示時，選擇 Yes, Delete (是，刪除)。
6. (選用) 如果您刪除步進擴展政策或簡易擴展政策，請執行下列動作來刪除與政策相關聯的 CloudWatch 警示。您可以略過這些子步驟來保留警示，以供日後使用。
 - a. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
 - b. 在導覽窗格上，選擇 Alarms (警示)。
 - c. 選擇警示 (例如，Step-Scaling-AlarmHigh-AddCapacity)，然後依序選擇 Action (動作)、Delete (刪除)。
 - d. 出現確認提示時，請選擇刪除。

取得 Auto Scaling 群組的擴展政策 (AWS CLI)

刪除擴展政策之前，請使用以下 [describe-policies](#) 命令來查看為該 Auto Scaling 群組建立了哪些擴展政策。您可以在刪除政策和 CloudWatch 警示時使用輸出。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
```

您可以使用 `--query` 參數，依擴展政策類型篩選結果。以下的 `query` 語法適用於 Linux 或 macOS。在 Windows 上，單引號需改為雙引號。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg  
--query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

以下為範例輸出。

```
[  
  {  
    "AutoScalingGroupName": "my-asg",  
    "PolicyName": "cpu50-target-tracking-scaling-policy",  
    "PolicyARN": "PolicyARN",  
    "PolicyType": "TargetTrackingScaling",  
    "StepAdjustments": [],  
    "Alarms": [  
      {  
        "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",  
        "AlarmName": "TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e"  
      },  
      {  
        "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",  
        "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2"  
      }  
    ],  
    "TargetTrackingConfiguration": {  
      "PredefinedMetricSpecification": {  
        "PredefinedMetricType": "ASGAverageCPUUtilization"  
      },  
      "TargetValue": 50.0,  
      "DisableScaleIn": false  
    },  
    "Enabled": true  
  }  
]
```

```
]
```

刪除擴展政策 (AWS CLI)

使用以下 [delete-policy](#) 命令。

```
aws autoscaling delete-policy --auto-scaling-group-name my-asg \  
  --policy-name cpu50-target-tracking-scaling-policy
```

刪除 CloudWatch 警示 (AWS CLI)

對於步進和簡易擴展政策，請使用 [delete-alarms](#) 命令來刪除與政策相關聯的 CloudWatch 警示。您可以略過此步驟來保留警示，以供日後使用。您可以同時刪除一或多個警示。例如，使用下列命令來刪除 Step-Scaling-AlarmHigh-AddCapacity 和 Step-Scaling-AlarmLow-RemoveCapacity 警示。

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-RemoveCapacity
```

的範例擴展政策 AWS CLI

您可以透過 AWS 管理主控台、AWS Command Line Interface (AWS CLI) 或 SDKs 建立 Amazon EC2 Auto Scaling 的擴展政策。

下列範例示範如何使用 [put-scaling-policy](#) 命令為 Amazon EC2 Auto Scaling 建立擴展政策。AWS CLI [put-scaling-policy](#) 將每個#####替換為自己的資訊。

若要開始使用 撰寫擴展政策 AWS CLI，請參閱 [目標追蹤擴展政策](#)和 中的簡介練習 [步進和簡易擴展政策](#)。

範例 1：套用具備預先定義指標規格的目標追蹤擴展政策

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \  
  --target-tracking-configuration file://config.json  
{  
  "TargetValue": 50.0,  
  "PredefinedMetricSpecification": {  
    "PredefinedMetricType": "ASGAverageCPUUtilization"  
  }  
}
```

```
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [PredefinedMetricSpecification](#)。

Note

如果檔案不在目前的目錄中，請輸入檔案的完整路徑。如需從檔案讀取 AWS CLI 參數值的詳細資訊，請參閱 AWS Command Line Interface 《使用者指南》中的 [從檔案載入 AWS CLI 參數](#)。

範例 2：套用具備自訂指標規格的目標追蹤擴展政策

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
    "MetricName": "MyBacklogPerInstance",
    "Namespace": "MyNamespace",
    "Dimensions": [{
      "Name": "MyOptionalMetricDimensionName",
      "Value": "MyOptionalMetricDimensionValue"
    }],
    "Statistic": "Average",
    "Unit": "None"
  }
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [CustomizedMetricSpecification](#)。

範例 3：只針對擴增套用目標追蹤擴展政策

```
aws autoscaling put-scaling-policy --policy-name alb1000-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 1000.0,
```

```

"PredefinedMetricSpecification": {
  "PredefinedMetricType": "ALBRequestCountPerTarget",
  "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-
group/943f017f100becff"
},
"DisableScaleIn": true
}

```

範例 4：針對擴增套用步進擴展政策

```

aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
MetricIntervalLowerBound=10.0,MetricIntervalUpperBound=20.0,ScalingAdjustment=10 \

MetricIntervalLowerBound=20.0,MetricIntervalUpperBound=30.0,ScalingAdjustment=20 \
  MetricIntervalLowerBound=30.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1

```

記錄政策的 Amazon Resource Name (ARN)。在建立 CloudWatch 警示時，您將需要 ARN。

範例 5：針對縮減套用步進擴展政策

```

aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2

```

記錄政策的 Amazon Resource Name (ARN)。在建立 CloudWatch 警示時，您將需要 ARN。

範例 6：針對擴增套用簡易擴展政策

```

aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity --min-adjustment-magnitude 2

```

記錄政策的 Amazon Resource Name (ARN)。在建立 CloudWatch 警示時，您將需要 ARN。

範例 7：針對縮減套用簡易擴展政策

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \  
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \  
  --adjustment-type ChangeInCapacity --cooldown 180
```

記錄政策的 Amazon Resource Name (ARN)。在建立 CloudWatch 警示時，您將需要 ARN。

Amazon EC2 Auto Scaling 的預測擴展

預測擴展的運作方式是分析歷史負載資料，以偵測流量中的每日或每週模式。它使用此資訊來預測未來容量需求，因此 Amazon EC2 Auto Scaling 可以主動增加 Auto Scaling 群組的容量，以符合預期的負載。

預測擴展非常適合以下情況：

- 循環流量，例如正常上班時間資源使用量大，夜間和週末資源使用量小
- 重複間歇工作負載模式，例如批次處理、測試或週期性資料分析
- 需要長時間才能初始化的應用程式，會在擴增事件期間對應用程式效能造成明顯的延遲影響

一般而言，如果有常規的流量增加模式，以及需要長時間才能初始化的應用程式，您應該考慮使用預測擴展。相較於僅使用動態擴展 (本質是被動式)，預測擴展可以在預測負載之前啟動容量，協助您更快擴展。預測擴展也可以協助您避免過度佈建容量，從而在 EC2 帳單上節省成本。

例如，考量應用程式在營業時間內具有高使用率而在夜間具有低使用率。在每個工作天開始時，預測擴展可以在第一次流量湧入之前新增容量。在從較低的使用率期間到較高的使用率期間時，這可協助您的應用程式維持高可用性和效能。您不必等待動態擴展來對不斷變化的流量做出反應。您也不必花時間檢閱應用程式的負載模式並嘗試使用排程的擴展來排程適當的容量。

主題

- [預測擴展的運作方式](#)
- [建立 Auto Scaling 群組的預測擴展政策](#)
- [評估您的預測擴展政策](#)
- [使用排程動作覆寫預測值](#)
- [使用自訂指標的進階預測擴展政策](#)

預測擴展的運作方式

本主題說明預測擴展的運作方式，並說明建立預測擴展政策時應考量的事項。

主題

- [運作方式](#)
- [容量上限](#)
- [考量事項](#)
- [支援的區域](#)

運作方式

若要使用預測擴展，請建立預測擴展政策，指定要監控和分析的 CloudWatch 指標。若要讓預測擴展開始預測未來值，此指標必須至少有 24 小時的資料。

建立政策後，預測性擴展會開始分析過去最多 14 天內的指標資料，以識別模式。它使用此分析來產生未來 48 小時容量需求的每小時預測。預測會使用最新的 CloudWatch 資料，每 6 小時更新一次。隨著新資料傳入，預測擴展能夠持續改善未來預測的準確性。

當您首次啟用預測性擴展時，它會以僅預測模式執行。在此模式中，它會產生容量預測，但不會根據這些預測實際擴展 Auto Scaling 群組。這可讓您評估預測的準確性和適用性。您可以使用 `GetPredictiveScalingForecast` API 操作或檢視預測資料 AWS 管理主控台。

在您檢閱預測資料並決定根據該資料開始擴展之後，請將擴展政策切換至預測和擴展模式。在此模式中：

- 如果預測預期負載增加，Amazon EC2 Auto Scaling 將透過向外擴展來增加容量。
- 如果預測預期負載減少，則不會縮減以移除容量。如果您想要移除不再需要的容量，您必須建立動態擴展政策。

根據預設，Amazon EC2 Auto Scaling 會根據該小時的預測，在每個小時開始時擴展 Auto Scaling 群組。您可以選擇使用 `PutScalingPolicy` API 操作中的 `SchedulingBufferTime` 屬性或中的啟動前執行個體設定來指定較早的開始時間 AWS 管理主控台。這會導致 Amazon EC2 Auto Scaling 在預測需求之前啟動新的執行個體，讓他們有時間開機並準備好處理流量。

為了支援在預測需求之前啟動新的執行個體，強烈建議您為 Auto Scaling 群組啟用預設執行個體暖機期。這會指定橫向擴展活動之後的時段，在此期間 Amazon EC2 Auto Scaling 不會縱向擴展，即使

動態擴展政策指出應減少容量。這可協助您確保新啟動的執行個體有足夠的時間開始為增加的流量提供服務，然後再考慮縮減操作。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

容量上限

Auto Scaling 群組具有最大容量設定，可限制群組可啟動的 EC2 執行個體數量上限。根據預設，當設定擴展政策時，它們不能將容量增加到高於其最大容量。

或者，您可以允許在預測容量接近或超過 Auto Scaling 群組的最大容量時，自動增加群組的最大容量。若要啟用此行為，請使用 PutScalingPolicy API 操作中的 MaxCapacityBreachBehavior 和 MaxCapacityBuffer 屬性，或 中的最大容量行為設定 AWS 管理主控台。

Warning

允許自動增加最大容量時請小心。如果未監控和管理增加的最大容量，這可能會導致啟動比預期更多的執行個體。增加的最大容量會成為 Auto Scaling 群組的新正常最大容量，直到您手動更新為止。最大容量不會自動減少回原始最大容量。

考量事項

- 確認預測擴展是否適合您的工作負載。如果工作負載顯示特定於星期幾或一天中時間的週期性負載模式，則工作負載非常適合預測擴展。若要檢查此項目，請在僅預測模式下設定預測擴展政策，然後參考主控台內的建議。Amazon EC2 Auto Scaling 會根據潛在政策效能的觀察提供建議。在讓預測擴展主動擴展應用程式之前，請先評估預測和建議。
- 預測擴展需要至少 24 小時的歷史資料才能開始預測。不過，如果歷史資料跨越整整兩週，那麼預測會更加有效。如果透過建立新的 Auto Scaling 群組並刪除舊群組來更新應用程式，則新的 Auto Scaling 群組需要 24 小時的歷史負載資料，之後預測擴展才能夠再次開始產生預測。您可以使用自訂指標來彙總所有新舊 Auto Scaling 群組中的指標。否則，您可能需要等待幾天才能獲得更準確的預測。
- 選擇能準確代表應用程式完整負載，且是應用程式最重要擴展方面的負載指標。
- 使用動態擴展搭配預測擴展可協助您密切遵循應用程式的需求曲線、在低流量期間縮減規模，以及在流量高於預期時擴展規模。當多個擴展政策處於作用中狀態時，每個政策會獨立決定所需的容量，並將所需容量設定為其中的最大值。例如，如果在目標追蹤擴展政策中需要 10 個執行個體維持在目標使用率，且在預測擴展政策中需要 8 個執行個體維持在目標使用率，則會將群組所需容量設定為 10。如果您是初次使用動態擴展，我們建議您使用目標追蹤擴展政策。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 動態擴展](#)。

- 預測性擴展的一個核心假設是 Auto Scaling 群組是同質的，並且所有執行個體的容量都相等。如果您的群組不是這樣，則預測容量可能不準確。因此，為[混合執行個體群組](#)建立預測擴展政策時請小心，因為可以佈建容量不相等的不同類型執行個體。以下是預測容量不準確的一些範例：
- 您的預測擴展政策以 CPU 使用率為基礎，但每個 Auto Scaling 執行個體上的 vCPU 數量會因執行個體類型而異。
- 您的預測擴展政策以網路輸入或網路輸出為基礎，但每個 Auto Scaling 執行個體的網路頻寬輸送量會因執行個體類型而異。例如，M5 與 M5n 執行個體類型相似，但 M5n 執行個體類型提供的網路輸送量明顯較高。

支援的區域

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太地區 (香港)
- 亞太區域 (海德拉巴)
- 亞太地區 (雅加達)
- 亞太地區 (墨爾本)
- 亞太地區 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太地區 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 加拿大西部 (卡加利)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)

- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)
- 歐洲 (蘇黎世)
- 以色列 (特拉維夫)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (美國東部)
- AWS GovCloud (美國西部)

建立 Auto Scaling 群組的預測擴展政策

下列程序可協助您使用 AWS 管理主控台 或 建立預測擴展政策 AWS CLI。

如果是新的 Auto Scaling 群組，則必須提供至少 24 小時的資料，然後 Amazon EC2 Auto Scaling 才能產生預測。

目錄

- [建立預測擴展政策 \(主控台\)](#)
- [建立預測擴展政策 \(AWS CLI\)](#)

建立預測擴展政策 (主控台)

如果這是您第一次建立預測擴展政策，我們建議您使用 主控台，在僅預測模式下建立多個預測擴展政策。這可讓您測試不同指標和目標值的潛在影響。您可以為每個 Auto Scaling 群組建立多個預測擴展政策，但只有其中一個政策可用於作用中擴展。

在主控台中建立預測擴展政策 (預先定義的指標)

利用預先定義的指標 (CPU、網路 I/O 或每個目標的 Application Load Balancer 請求計數)，使用以下程序來建立預測擴展政策。建立預測擴展政策最簡單的方法是使用預先定義的指標。但如果您偏好使用自訂指標，請參閱 [在主控台中建立預測擴展政策 \(自訂指標\)](#)。

建立預測擴展政策

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Automatic scaling (自動擴展) 索引標籤的 Scaling policies (擴展政策) 中，選擇 Create predictive scaling policy (建立預測擴展政策)。
4. 輸入政策的名稱。
5. 開啟 Scale based on forecast (依據預測擴展)，授予 Amazon EC2 Auto Scaling 許可立即開始擴展。

若要將政策保留在 Forecast only (僅預測) 模式，保持 Scale based on forecast (依據預測擴展) 為關閉。

6. 針對 Metrics (指標)，從選項清單中選擇您的指標。選項包括 CPU、Network In (網路輸入)、Network Out (網路輸出)、Application Load Balancer request count (Application Load Balancer 請求計數)，以及 Custom metric pair (自訂指標對)。

如果選擇了 Application Load Balancer request count per target (每個目標的 Application Load Balancer 請求計數)，則在 Target group (目標群組) 中選擇目標群組。只有將 Application Load Balancer 目標群組連接至 Auto Scaling 群組後，才支援 Application Load Balancer request count per target (每個目標的 Application Load Balancer 請求計數)。

如果選擇了 Custom metric pair (自訂指標對)，則從 Load metric (負載指標) 和 Scaling metric (擴展指標) 下拉式清單中選擇個別指標。

7. 對於 Target utilization (目標使用率)，輸入 Amazon EC2 Auto Scaling 應維護的目標值。Amazon EC2 Auto Scaling 會擴增容量，直到平均使用率達到目標使用率，或直到達到所指定執行個體數目上限為止。

如果擴展指標為...	則目標使用率代表...
CPU	每個執行個體在理想情況下應該使用的 CPU 百分比。
網路傳入	每個執行個體在理想情況下應該接收的每分鐘平均位元組數量。

如果擴展指標為...	則目標使用率代表...
網路輸出	每個執行個體在理想情況下應該傳出的每分鐘平均位元組數量。
每個目標的 Application Load Balancer 請求計數	每個執行個體在理想情況下應該接收的每分鐘平均請求數量。

8. (選用) 對於 Pre-launch instances (預先啟動執行個體)，選擇您希望預測呼叫增加負載之前啟動執行個體的時間。
9. (選用) 對於 Max capacity behavior (最大容量行為)，在預測容量超過定義的最大容量時，選擇是否讓 Amazon EC2 Auto Scaling 擴展至高於群組的最大容量。開啟此設定讓您能在預測流量達到最高值時擴展容量。
10. (選用) 對於 Buffer maximum capacity above the forecasted capacity (緩衝區最大容量高於預測容量)，選擇預測容量接近或超過容量上限時要使用多少額外容量。指定數值是相對於預測容量的百分比。例如，如果緩衝區為 10，這表示 10% 的緩衝區。因此，如果預測容量為 50 而最大容量為 40，則有效最大容量是 55。

如果設定為 0，Amazon EC2 Auto Scaling 可以將容量擴展至高於最大容量，以等於但不超過預測容量。

11. 選擇建立預測擴展政策。

在主控台中建立預測擴展政策 (自訂指標)

利用自訂指標，使用以下程序來建立預測擴展政策。自訂指標可以包含 CloudWatch 提供的其他指標或是您發佈至 CloudWatch 的指標。若要使用 CPU、網路 I/O 或每個目標的 Application Load Balancer 請求計數，請參閱 [在主控台中建立預測擴展政策 \(預先定義的指標\)](#)。

若要利用自訂指標建立預測擴展政策，您必須執行以下操作：

- 您必須提供可讓 Amazon EC2 Auto Scaling 與 CloudWatch 中的指標互動的原始查詢。如需詳細資訊，請參閱[使用自訂指標的進階預測擴展政策](#)。為確保 Amazon EC2 Auto Scaling 能夠從 CloudWatch 擷取指標資料，請確認每個查詢都有傳回資料點。確認的方法為使用 CloudWatch 主控台或 CloudWatch [GetMetricData](#) API 操作。

Note

我們在 Amazon EC2 Auto Scaling 主控台的 JSON 編輯器中提供範例 JSON 承載。這些範例提供金鑰值對的參考，這些值對是新增由提供的其他 CloudWatch 指標 AWS 或您先前發佈至 CloudWatch 的指標所需的。您能以這些範例為起點，然後根據需要加以自訂。

- 如果您使用任何指標數學，則必須手動建構 JSON 以符合您的獨特情境。如需詳細資訊，請參閱[使用指標數學表達式](#)。在於政策中使用指標數學之前，應先確認以指標數學表達式為基礎的指標查詢是否有效，且會傳回單一的時間序列。確認的方法為使用 CloudWatch 主控台或 CloudWatch [GetMetricData](#) API 操作。

如果因為提供錯誤的資料 (例如錯誤的 Auto Scaling 群組名稱) 而在查詢中發生錯誤，則預測將不會有任何資料。如需自訂指標問題的疑難排解，請參閱[預測擴展政策中自訂指標的考量事項](#)。

建立預測擴展政策

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Automatic scaling (自動擴展) 索引標籤的 Scaling policies (擴展政策) 中，選擇 Create predictive scaling policy (建立預測擴展政策)。
4. 輸入政策的名稱。
5. 開啟 Scale based on forecast (依據預測擴展)，授予 Amazon EC2 Auto Scaling 許可立即開始擴展。

若要將政策保留在 Forecast only (僅預測) 模式，保持 Scale based on forecast (依據預測擴展) 為關閉。

6. 針對 Metrics (指標)，選擇 Custom metric pair (自訂指標對)。
 - a. 針對 Load metric (負載指標)，選擇 Custom CloudWatch metric (自訂 CloudWatch 指標) 以使用自訂指標。建構含有政策負載指標定義的 JSON 承載，並將其貼到 JSON 編輯器框中，替換框中已有的內容。

- b. 針對 Scaling metric (擴展指標)，選擇 Custom CloudWatch metric (自訂 CloudWatch 指標) 以使用自訂指標。建構含有政策擴展指標定義的 JSON 承載，並將其貼到 JSON 編輯器框中，替換框中已有的內容。
- c. (選用) 若要新增自訂容量指標，請選取 Add custom capacity metric (新增自訂容量指標) 的核取方塊。建構含有政策容量指標定義的 JSON 承載，並將其貼到 JSON 編輯器框中，替換框中已有的內容。

如果容量指標資料橫跨多個 Auto Scaling 群組，您只需啟用此選項即可為容量建立新的時間序列。在此情況下，您必須使用指標數學將資料彙總為單一時間序列。

7. 對於 Target utilization (目標使用率)，輸入 Amazon EC2 Auto Scaling 應維護的目標值。Amazon EC2 Auto Scaling 會擴增容量，直到平均使用率達到目標使用率，或直到達到所指定執行個體數目上限為止。
8. (選用) 對於 Pre-launch instances (預先啟動執行個體)，選擇您希望預測呼叫增加負載之前啟動執行個體的間隔時間。
9. (選用) 對於 Max capacity behavior (最大容量行為)，在預測容量超過定義的最大容量時，選擇是否讓 Amazon EC2 Auto Scaling 擴展至高於群組的最大容量。開啟此設定讓您能在預測流量達到最高值時擴展容量。
10. (選用) 對於 Buffer maximum capacity above the forecasted capacity (緩衝區最大容量高於預測容量)，選擇預測容量接近或超過容量上限時要使用多少額外容量。指定數值是相對於預測容量的百分比。例如，如果緩衝區為 10，這表示 10% 的緩衝區。因此，如果預測容量為 50 而最大容量為 40，則有效最大容量是 55。

如果設定為 0，Amazon EC2 Auto Scaling 可以將容量擴展至高於最大容量，以等於但不超過預測容量。

11. 選擇建立預測擴展政策。

建立預測擴展政策 (AWS CLI)

使用 AWS CLI 設定 Auto Scaling 群組的預測擴展政策，如下所示。將每個 `#####` 替換為自己的資訊。

如需有關可指定之 CloudWatch 指標的詳細資訊，請參閱 Amazon EC2 Auto Scaling API Reference 中的 [PredictiveScalingMetricSpecification](#)。

範例 1：建立預測但不擴展的預測擴展政策

下列範例政策顯示完整的政策組態，該組態會使用 CPU 使用率指標進行預測擴展 (目標使用率為 40)。除非您明確指定要使用的模式，否則依據預設會使用 ForecastOnly 模式。將此組態儲存至名為 config.json 的檔案。

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 40,
      "PredefinedMetricPairSpecification": {
        "PredefinedMetricType": "ASGCPUtilization"
      }
    }
  ]
}
```

若要從命令列建立政策，請執行 [put-scaling-policy](#) 命令搭配指定的組態檔案，如以下範例所示。

```
aws autoscaling put-scaling-policy --policy-name cpu40-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令會傳回政策的 Amazon Resource Name (ARN)。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/cpu40-predictive-scaling-policy",
  "Alarms": []
}
```

範例 2：預測並擴展的預測擴展政策

對於允許 Amazon EC2 Auto Scaling 預測和擴展的政策，新增值為 ForecastAndScale 的屬性 Mode。下列範例顯示使用 Application Load Balancer 請求計數指標的政策組態。目標使用率為 1000，且預測擴展會設定為 ForecastAndScale 模式。

```
{
  "MetricSpecifications": [
```

```
{
  "TargetValue": 1000,
  "PredefinedMetricPairSpecification": {
    "PredefinedMetricType": "ALBRequestCount",
    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
  }
},
"Mode": "ForecastAndScale"
}
```

若要建立此政策，請執行 [put-scaling-policy](#) 命令與指定的組態檔案，如下列範例展示的方式所示。

```
aws autoscaling put-scaling-policy --policy-name alb1000-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令會傳回政策的 Amazon Resource Name (ARN)。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-
id:scalingPolicy:19556d63-7914-4997-8c81-d27ca5241386:autoScalingGroupName/my-
asg:policyName/alb1000-predictive-scaling-policy",
  "Alarms": []
}
```

範例 3：可擴展為高於容量上限的預測擴展政策

下列範例顯示當您需要處理高於正常負載時，如何建立可以擴展為高於群組大小上限的政策。根據預設，Amazon EC2 Auto Scaling 不會將 EC2 容量擴展到高於您定義的最大容量。然而，以稍微多一些容量讓它擴展得稍高一些，有助於避免效能或可用性問題。

若要為 Amazon EC2 Auto Scaling 提供空間，以便在將容量預測為或非常接近群組的大小上限時佈建額外容量，請指定 `MaxCapacityBreachBehavior` 和 `MaxCapacityBuffer` 屬性，如下列範例所示。您必須指定數值為 `IncreaseMaxCapacity` 的 `MaxCapacityBreachBehavior`。群組可擁有的執行個體的最大數量取決於 `MaxCapacityBuffer` 的數值。

```
{
  "MetricSpecifications": [
    {
```



```
        "TargetValue": 70,
        "PredefinedMetricPairSpecification": {
            "PredefinedMetricType": "ASGCPUUtilization"
        }
    },
    "MaxCapacityBreachBehavior": "IncreaseMaxCapacity",
    "MaxCapacityBuffer": 10
}
```

在此範例中，政策設定為使用 10% 的緩衝區 ("MaxCapacityBuffer": 10)，因此如果預測容量為 50 而最大容量為 40，則有效的最大容量是 55。如果政策可將容量擴展至高於容量上限，以便等於但不超過預測容量，則緩衝區為 0 ("MaxCapacityBuffer": 0)。

若要建立此政策，請執行 [put-scaling-policy](#) 命令與指定的組態檔案，如下列範例展示的方式所示。

```
aws autoscaling put-scaling-policy --policy-name cpu70-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令會傳回政策的 Amazon Resource Name (ARN)。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:d02ef525-8651-4314-
bf14-888331ebd04f:autoScalingGroupName/my-asg:policyName/cpu70-predictive-scaling-
policy",
  "Alarms": []
}
```

評估您的預測擴展政策

使用預測擴展政策來擴展 Auto Scaling 群組之前，請在 Amazon EC2 Auto Scaling 主控台中檢閱建議和政策的其他資料。這很重要，因為您不希望預測擴展政策在您知道其預測準確之前擴展實際容量。

如果 Auto Scaling 群組是新群組，Amazon EC2 Auto Scaling 需要 24 小時才能建立第一個預測。

Amazon EC2 Auto Scaling 建立預測時，會使用歷史資料。如果您的 Auto Scaling 群組還沒有太多的最新歷史資料，Amazon EC2 Auto Scaling 可能會使用從目前可用的歷史彙總建立的彙總來暫時回填預測。預測會在政策建立日期前的兩週內回填。

目錄

- [檢視您的預測擴展建議](#)
- [檢閱預測擴展監控圖表](#)
- [使用 CloudWatch 監控預測擴展指標](#)

檢視您的預測擴展建議

為了獲得有效的分析，Amazon EC2 Auto Scaling 應具有至少兩個可進行比較的預測擴展政策。(不過，您仍然可以檢閱單一政策的問題清單。) 建立多個政策時，您可以根據使用不同指標的政策，評估使用一個指標的政策。您也可以評估不同目標值和指標組合的影響。建立預測擴展政策之後，Amazon EC2 Auto Scaling 會立即開始評估哪些政策可以更好地擴展您的群組。

在 Amazon EC2 Auto Scaling 主控台中檢視建議

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Auto Scaling 索引標籤的預測擴展政策下，您可以檢視政策的相關詳細資訊以及我們的建議。該建議會告訴您使用預測擴展政策的結果是否優於不使用它。

如果您不確定預測擴展政策是否適合您的群組，請檢閱可用性影響和成本影響欄，以選擇正確的政策。每一欄的資訊都會說明政策的影響。

- 可用性影響：說明政策是否會佈建足夠的執行個體來處理工作負載，以避免對可用性造成負面影響 (相較於不使用該政策)。
- 成本影響：說明政策是否不會過度佈建執行個體，以避免對成本造成負面影響 (相較於不使用該政策)。如果過度佈建比較嚴重，執行個體就會出現使用率過低或閒置情況，這只會增加成本影響。

如果您有多個政策，則以較低成本提供最多可用性優勢的政策名稱旁會顯示最佳預測標籤。可用性影響會獲得更多加權。

4. (選用) 若要選取建議結果所需的期間，請從評估期間下拉式清單中選擇您偏好的值：2 天、1 週、2 週、4 週、6 週或 8 週。根據預設，評估期間是最近兩週。較長的評估期間會為建議結果提供更多資料點。不過，如果負載模式發生變更 (例如在一段異常需求期間後)，新增更多資料點可能無法改善結果。在這種情況下，您可以查看最新資料以獲得更有針對性的建議。

Note

只會針對處於僅預測模式的政策產生建議。當政策在整個評估期間都處於僅預測模式時，建議功能的效果會更好。如果您在預測和擴展模式中啟動政策，並於稍後將其切換至僅預測模式，則該政策的問題清單可能會有偏差。這是因為該政策已經為實際容量做出了貢獻。

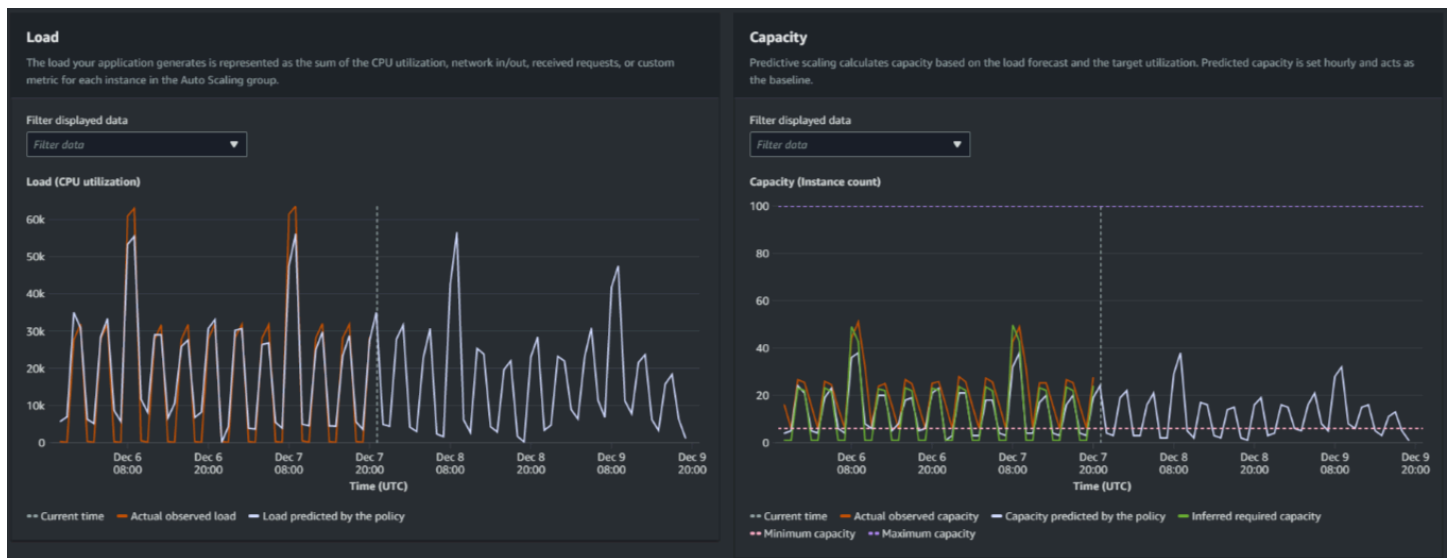
檢閱預測擴展監控圖表

在 Amazon EC2 Auto Scaling 主控台中，您可以檢閱前幾天、前幾週或前幾個月的預測，以視覺化方式呈現政策在一段時間內的表現。在決定是否允許政策擴展實際容量時，您也可以使用此資訊來評估預測的準確性。

在 Amazon EC2 Auto Scaling 主控台中檢閱預測擴展監控圖表

1. 從預測擴展政策清單中選擇政策。
2. 在監控區段中，您可以根據實際值檢視政策在過去和未來的負載和容量預測。負載圖表會顯示所選負載指標的負載預測與實際值。容量圖表會顯示政策預測的執行個體數量。它還包括實際啟動的執行個體數量。垂直線會將歷史值與未來預測隔開。建立政策後，這些圖表很快就可以使用。
3. (選用) 若要變更圖表中顯示的歷史資料量，請從頁面頂端的評估期間下拉式清單中選擇您偏好的值。評估期間不會以任何方式轉換此頁面上的資料。它只會變更顯示的歷史資料量。

下圖顯示已多次套用預測時的負載和容量圖表。預測擴展會根據您的歷史負載資料預測負載。應用程式產生的負載會以 Auto Scaling 群組中每個執行個體的 CPU 使用率、網路輸入/輸出、收到的請求或自訂指標的總和表示。預測擴展會根據您希望為擴展指標達成的負載預測和目標使用率來計算未來容量需求。



比較負載圖表中的資料

每條水平線代表每間隔一小時報告的一組不同資料點：

1. 實際觀察到的負載會使用所選負載指標的 SUM 統計資料來顯示過去的每小時總負載。
2. 政策預測的負載會顯示每小時的負載預測。此預測是基於前兩週的實際負載觀察結果。

比較容量圖表中的資料

每條水平線代表每間隔一小時報告的一組不同資料點：

1. 實際觀察到的容量會顯示啟用 [GroupTotalInstances](#) 指標時 Auto Scaling 群組的過去實際容量。此容量取決於您的其他擴展政策和所選期間內的群組大小下限。
2. 政策預測的容量會顯示政策處於預測和擴展模式時，可預期在每小時開始時獲得的基準容量。
3. 推斷的所需容量會顯示將擴展指標維持在所選目標值的理想容量。
4. 最小容量會顯示 Auto Scaling 群組的最小容量。
5. 最大容量會顯示 Auto Scaling 群組的最大容量。

為了計算推斷的所需容量，我們首先假設以指定的目標值平均使用每個執行個體。實際上，並不會平均使用執行個體。但是，假設使用率均勻地分散在執行個體之間，我們就可以對所需容量進行可能的估算。然後，容量需求的計算結果會與您用於預測擴展政策的擴展指標成反比。換句話說，隨著容量增加，擴展指標會以相同的速率減少。例如，如果容量加倍，擴展指標必定會減半。

推斷的所需容量公式：

$$\text{sum of } (\text{actualCapacityUnits} * \text{scalingMetricValue}) / (\text{targetUtilization})$$

例如，我們使用特定一小時的 `actualCapacityUnits` (10) 和 `scalingMetricValue` (30)。然後，我們會使用您在預測擴展政策中指定的 `targetUtilization` (60)，並計算同一小時內推斷的所需容量。這會傳回值 5。這表示 5 是維持容量與擴展指標目標值正好成反比所需的推斷容量。

Note

您可以使用各種控制桿來調整和改善應用程式的成本節省效益和可用性。

- 您可以針對基準容量使用預測擴展，並使用動態擴展來處理額外的容量。動態擴展會與預測擴展分開運作，可根據目前的使用率進行縮減和擴增。首先，Amazon EC2 Auto Scaling 會針對每個動態擴展政策計算建議的執行個體數量。然後，它會根據提供最多執行個體數量的政策進行擴展。
- 為了允許在負載減少時進行縮減，Auto Scaling 群組應隨時至少具有一個動態擴展政策，並啟用縮減部分。
- 您可以確保您的最小和最大容量沒有太大限制，以提高擴展效能。如果政策的建議執行個體數量不在最小和最大容量範圍內，將無法進行縮減和擴增。

使用 CloudWatch 監控預測擴展指標

根據需求，您可能偏好從 Amazon CloudWatch 而非 Amazon EC2 Auto Scaling 主控台中存取預測擴展的監控資料。建立預測擴展政策之後，政策會收集用於預測未來負載和容量的資料。收集這些資料後，資料會以固定間隔定期自動儲存在 CloudWatch 中。然後，您可以使用 CloudWatch 以視覺化方式呈現政策在一段時間內的表現。您也可以建立 CloudWatch 警示，以便在效能指標變更超出您在 CloudWatch 中定義的限制時通知您。

主題

- [視覺化歷史預測資料](#)
- [使用指標數學建立準確度指標](#)

視覺化歷史預測資料

您可以在 CloudWatch 中檢視預測擴展政策的負載和容量預測資料。在單一圖表中根據其他 CloudWatch 指標視覺化預測時，此功能非常有用。它還可以幫助您檢視更廣泛的時間範圍，以便可以看到隨時間變化的趨勢。您可以存取長達 15 個月的歷史指標，以更加了解政策的執行狀況。

如需詳細資訊，請參閱[預測擴展指標與維度](#)。

使用 CloudWatch 主控台檢視歷史預測資料

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。
3. 選擇 Auto Scaling (自動擴展) 指標命名空間。
4. 選擇下列其中一個選項，以檢視負載預測或容量預測指標：
 - 預測性擴展載入預測
 - 預測性擴展容量預測
5. 在搜尋欄位中，輸入預測擴展政策的名稱或者 Auto Scaling 群組的名稱，然後按 Enter 篩選結果。
6. 若要將指標圖形化，請勾選指標旁的核取方塊。若要變更圖形的名稱，請選擇鉛筆圖示。若要變更時間範圍，請選取一個預先定義的值，或選擇 custom (自訂)。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[建立指標圖表](#)。
7. 若要變更統計數字，請選擇 Graphed metrics (圖表化指標) 索引標籤。選擇欄位標題或個別的值，然後選擇不同的統計資料。雖然您可以為每個指標選擇任何統計資料，但並非所有統計資料都適用於 PredictiveScalingLoadForecast 和 PredictiveScalingCapacityForecast 指標。例如，Average (平均值)、Minimum (最小值) 和 Maximum (最大值) 統計資料有用，但是 Sum (總和) 統計資料無用。
8. 若要將其他指標新增到圖表，請在 Browse (瀏覽) 下，選擇 All (所有)，找到特定指標，然後選取旁邊的核取方塊。您最多可新增 10 個指標。

例如，若要將 CPU 使用率的實際值新增至圖表，請選擇 EC2 命名空間，然後選擇 By Auto Scaling Group (依據 Auto Scaling 群組)。然後，選取 CPUUtilization (CPU 使用率) 指標的核取方塊和特定 Auto Scaling 群組。

9. (選用) 若要將圖表新增至 CloudWatch 儀表板，請選擇 Actions (動作)、Add to dashboard (新增至儀表板)。

使用指標數學建立準確度指標

利用指標數學，可查詢多個 CloudWatch 指標，並使用數學表達式根據這些指標來建立新的時間序列。您可以在 CloudWatch 主控台視覺化產生的時間序列，並將其新增至儀表板。如需有關指標數學的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[使用指標數學](#)。

使用指標數學，您可以使用不同的方式繪製 Amazon EC2 Auto Scaling 為預測擴展而產生的資料圖表。這可協助您監控一段時間內的政策績效，並協助您瞭解是否可以改善指標組合。

例如，您可以使用指標數學表達式來監控[平均絕對誤差百分比](#) (MAPE)。MAPE 指標有助於監控預測值與指定預測期間觀察到的實際值之間的差異。MAPE 值的變化可指示政策的績效是否隨著應用程式性質的變化在一段時間內而降低。MAPE 的增加表示預測值與實際值之間的差距更大。

範例：指標數學表達式

如果要開始使用此類圖表，您可以建立類似於以下範例中所示的指標數學表達式。

```
{
  "MetricDataQueries": [
    {
      "Expression": "TIME_SERIES(AVG(ABS(m1-m2)/m1))",
      "Id": "e1",
      "Period": 3600,
      "Label": "MeanAbsolutePercentageError",
      "ReturnData": true
    },
    {
      "Id": "m1",
      "Label": "ActualLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/EC2",
          "MetricName": "CPUUtilization",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            }
          ]
        },
        "Period": 3600,
        "Stat": "Sum"
      },
      "ReturnData": false
    },
    {
      "Id": "m2",
      "Label": "ForecastedLoadValues",
      "MetricStat": {
```



```

    "Metric": {
      "Namespace": "AWS/AutoScaling",
      "MetricName": "PredictiveScalingLoadForecast",
      "Dimensions": [
        {
          "Name": "AutoScalingGroupName",
          "Value": "my-asg"
        },
        {
          "Name": "PolicyName",
          "Value": "my-predictive-scaling-policy"
        },
        {
          "Name": "PairIndex",
          "Value": "0"
        }
      ]
    },
    "Period": 3600,
    "Stat": "Average"
  },
  "ReturnData": false
}
]
}

```

`MetricDataQueries` 有一個指標資料查詢結構陣列,而不是單一的指標。`MetricDataQueries` 中的每個項目都會取得指標或執行數學表達式。第一項, `e1`, 是數學表達式。指定的表達式將 `ReturnData` 參數設定為 `true`, 最終產生單一時間序列。對於所有其他指標, `ReturnData` 值為 `false`。

在此範例中, 指定的表達式會使用實際值和預測值作為輸入值, 並傳回新指標 (MAPE)。m1 是包含實際負載值的 CloudWatch 指標 (假設 CPU 使用率是針對名為 `my-predictive-scaling-policy` 的政策最初指定的負載指標)。m2 是包含預測負載值的 CloudWatch 指標。MAPE 指標的數學語法如下：

Average of $(\text{abs}((\text{Actual} - \text{Forecast})/(\text{Actual})))$

視覺化您的準確度指標並設定警示

若要視覺化準確度指標資料, 請選取 CloudWatch 主控台中的 Metrics (指標) 索引標籤。您可以從那裡繪製資料圖表。如需詳細資訊, 請參閱《Amazon CloudWatch 使用者指南》中的 [將數學表達式新增到 CloudWatch 圖表](#)。

您也可以在 Metrics (指標) 區段中，對您監控的指標設定警示。在 Graphed metrics (圖表化指標) 索引標籤上，選取 Actions (動作) 資料欄下的 Create alarm (建立警示) 圖示。Create alarm (建立警示) 圖示表示為一個小鐘。如需詳細資訊與通知選項，請參閱 Amazon CloudWatch User Guide 中的 [Creating a CloudWatch alarm based on a metric math expression](#) 與 [Notifying users on alarm changes](#)。

或者，您可以使用 [GetMetricData](#) 和 [PutMetricAlarm](#) 利用指標數學來執行計算，並根據輸出建立警示。

使用排程動作覆寫預測值

有時候，您可能會有未來應用程式需求的其他資訊，但預測計算無法考量該資訊。例如，預測計算可能會低估即將到來的行銷活動所需的容量。您可以使用排程動作，在未來時段暫時覆寫預測。排程動作可以週期性執行，或在一次性需求波動的特定日期與時間執行。

例如，您可以建立最小容量高於預測容量的排程動作。在執行時間，Amazon EC2 Auto Scaling 會更新 Auto Scaling 群組的最小容量。由於預測擴展會針對容量進行最佳化，因此會接受容量下限高於預測值的排程動作。這樣可以防止容量低於預期。若要停止複寫預測，請使用第二個排程動作，讓最小容量恢復至其原始設定。

下列程序概述在未來時段覆寫預測的步驟。

主題

- [步驟 1：\(選用\) 分析時間序列資料](#)
- [步驟 2：建立兩個排程動作](#)

Important

本主題假設您正嘗試覆寫預測，以擴展至高於預測值的容量。如果您需要暫時減少容量，而不會受到預測擴展政策的干擾，請改用僅限預測模式。在僅限預測模式下，預測擴展將繼續產生預測，但不會自動增加容量。然後，您可以監控資源使用率，並視需要手動減少群組的大小。如需手動擴展的詳細資訊，請參閱[Amazon EC2 Auto Scaling 手動擴展](#)。

步驟 1：(選用) 分析時間序列資料

從分析預測時間序列資料開始。這是選用步驟，但是如果您想要了解預測的詳細資訊，這會很有幫助。

1. 擷取預測

建立預測之後，您可以查詢預測中的特定時段。查詢的目標是取得特定時段的時間序列資料的完整檢視。

查詢最多可包含未來兩天的預測資料。如果已經使用預測擴展一段時間，您也可以存取過去的預測資料。不過，開始和結束時間之間的最大持續時間是 30 天。

若要使用 `get-predictive-scaling-forecast` AWS CLI 命令取得預測，請在 命令中提供下列參數：

- 在 `--auto-scaling-group-name` 參數中輸入 Auto Scaling 群組名稱。
- 在 `--policy-name` 參數中輸入政策名稱。
- 在 `--start-time` 參數中輸入開始時間，以便僅將指定時間之時或之後的預測資料傳回。
- 在 `--end-time` 參數中輸入結束時間，以便僅將指定時間之前的預測資料傳回。

```
aws autoscaling get-predictive-scaling-forecast --auto-scaling-group-name my-asg \
--policy-name cpu40-predictive-scaling-policy \
--start-time "2021-05-19T17:00:00Z" \
--end-time "2021-05-19T23:00:00Z"
```

如果成功，此命令會傳回類似下方範例的資料。

```
{
  "LoadForecast": [
    {
      "Timestamps": [
        "2021-05-19T17:00:00+00:00",
        "2021-05-19T18:00:00+00:00",
        "2021-05-19T19:00:00+00:00",
        "2021-05-19T20:00:00+00:00",
        "2021-05-19T21:00:00+00:00",
        "2021-05-19T22:00:00+00:00",
        "2021-05-19T23:00:00+00:00"
      ],
      "Values": [
        153.0655799339254,
        128.8288551285919,
        107.1179447150675,
        197.3601844551528,
        626.4039934516954,
        596.9441277518481,

```

```

        677.9675713779869
      ],
      "MetricSpecification": {
        "TargetValue": 40.0,
        "PredefinedMetricPairSpecification": {
          "PredefinedMetricType": "ASGCPUUtilization"
        }
      }
    ],
    "CapacityForecast": {
      "Timestamps": [
        "2021-05-19T17:00:00+00:00",
        "2021-05-19T18:00:00+00:00",
        "2021-05-19T19:00:00+00:00",
        "2021-05-19T20:00:00+00:00",
        "2021-05-19T21:00:00+00:00",
        "2021-05-19T22:00:00+00:00",
        "2021-05-19T23:00:00+00:00"
      ],
      "Values": [
        2.0,
        2.0,
        2.0,
        2.0,
        4.0,
        4.0,
        4.0
      ]
    },
    "UpdateTime": "2021-05-19T01:52:50.118000+00:00"
  }
}

```

回應包含兩種預測：LoadForecast 和 CapacityForecast。LoadForecast 顯示每小時負載預測。CapacityForecast 顯示每小時處理預測負載時所需的容量預測值，同時維持 TargetValue 為 40.0 (40% 的 CPU 平均使用率)。

2. 確定目標時段

確定應發生一次性需求波動時的一個小時或數個小時。請記住，預測中顯示的日期和時間為 UTC 格式。

步驟 2：建立兩個排程動作

接下來，在應用程式具有高於預測的負載時，為特定時段建立兩個排程動作。舉例來說，如果行銷活動會在特定時段為網站帶來流量，您可以排程一次性動作，在它開始時更新最小容量。然後，排程另一個動作，以便在事件結束時將最小容量恢復至原始設定。

為一次性事件建立兩個排程動作 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Automatic scaling (自動擴展) 索引標籤的 Scheduled actions (排程動作) 中，選擇 Create scheduled action (建立排程動作)。
4. 請填寫以下排定動作設定：
 - a. 輸入排程動作的 Name (名稱)。
 - b. 對於 Min (最小值)，輸入 Auto Scaling 群組的新最小容量。Min (最小值) 必須小於或等於該群組的大小上限。如果 Min (最小值) 的數值大於群組的大小上限，則必須更新 Max (最大值)。
 - c. 針對 Recurrence (重複)，選擇 Once (一次)。
 - d. 對於 Time zone (時區)，選擇時區。如果沒有選擇時區，預設會使用 ETC/UTC。
 - e. 定義 Specific start time (特定開始時間)。
5. 選擇建立。

主控台會顯示 Auto Scaling 群組的排程動作。

6. 設定第二個排程動作，以便在事件結束時將最小容量恢復至原始設定。預測擴展只有在您為 Min (最小值) 設定的數值低於預測值時才能擴展容量。

為一次性事件建立兩個排程動作 (AWS CLI)

若要使用 AWS CLI 建立排程動作，請使用 [put-scheduled-update-group-action](#) 命令。

例如，我們定義一個排程，在 5 月 19 日下午 5 點維持三個執行個體的最小容量，持續 8 小時。下列命令顯示如何實作此案例。

第一個 [put-scheduled-update-group-action](#) 命令會指示 Amazon EC2 Auto Scaling 在 2021 年 5 月 19 日下午 5 點 (UTC) 更新 Auto Scaling 群組的最小容量。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-start \  
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-  
capacity 3
```

第二個命令指示 Amazon EC2 Auto Scaling 在 2021 年 5 月 20 日上午 1 點 (UTC)，將群組的最小容量設定為 1。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end \  
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-  
capacity 1
```

再將這些排程動作新增到 Auto Scaling 群組後，Amazon EC2 Auto Scaling 會執行下列動作：

- 在 2021 年 5 月 19 日下午 5 點 (UTC)，第一個排程動作會執行。如果群組目前擁有少於 3 個執行個體，群組則會擴增至 3 個執行個體。在此時間和接下來的八個小時內，如果預測容量高於實際容量，或者如果動態擴展政策生效，則 Amazon EC2 Auto Scaling 可以繼續擴增。
- 在 2021 年 5 月 20 日上午 1 點 (UTC)，第二個排程動作會執行。這會在事件結束時將最小容量恢復至原始設定。

根據週期性排程擴展

若要每週覆寫相同時段的預測，請建立兩個排程動作，並使用 Cron 表達式提供時間與日期邏輯。

Cron 表達式格式由 5 個以空格分隔的欄位組成：[分鐘] [小時] [一個月的第幾日] [一年的第幾個月] [一週的第幾日]。欄位可以包含任何允許的數值，包括特殊字元。

例如，以下 Cron 表達式會在每周二上午 6:30 執行動作。使用星號作為萬用字元，以比對欄位的所有數值。

```
30 6 * * 2
```

另請參閱

如需如何建立、列出、編輯及刪除排程動作的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 排程擴展](#)。

使用自訂指標的進階預測擴展政策

在預測擴展政策中，您可以使用預先定義或自訂的指標。當預先定義指標 (CPU、網路輸入/輸出和 Application Load Balancer 請求計數) 未充分描述您的應用程式負載時，自訂指標會非常實用。

使用自訂指標建立預測擴展政策時，您可以指定提供的其他 CloudWatch 指標 AWS，也可以指定您自己定義和發佈的指標。您也可以使用指標數學來彙總現有指標，並將其轉換為 AWS 不會自動追蹤的新時間序列。當您合併資料中的值時 (例如，透過計算新的總和或平均值)，它稱為執行彙總。產生的資料稱為彙總。

下一節包含如何建構政策的 JSON 結構的最佳實務和範例。

主題

- [最佳實務](#)
- [先決條件](#)
- [建構自訂指標的 JSON](#)
- [預測擴展政策中自訂指標的考量事項](#)
- [限制](#)

最佳實務

以下最佳實務可協助您更有效地使用自訂指標：

- 對於負載規範，最實用的指標是以 Auto Scaling 群組作為整體 (而不考慮該群組的容量) 來表示負載的指標。
- 對於擴展指標規範，要擴展的最實用指標是每個執行個體指標的平均輸送量或使用率。
- 擴展指標必須與容量成反比。也就是說，如果 Auto Scaling 群組中的執行個體數量增加，則擴展指標應減少大致相同的比例。為確保預測擴展按預期進行，負載指標和擴展指標還必須彼此密切關聯。
- 目標使用率必須與擴展指標的類型相符。對於使用 CPU 使用率的政策組態，這是一個目標百分比。對於使用輸送量 (如請求或訊息數量) 的政策組態，這是在任何一分鐘間隔內每個執行個體的請求或訊息的目標數量。
- 如果不遵循這些建議，則時間序列的預測未來值可能不正確。要驗證資料是否正確，您可以在 Amazon EC2 Auto Scaling 主控台中檢視預測值。或者，在建立預測擴展政策後，檢查 [GetPredictiveScalingForecast](#) API 呼叫傳回的 LoadForecast 和 CapacityForecast 物件。
- 強烈建議您在 forecast only (僅預測) 模式中設定預測擴展，以便在預測擴展主動擴展容量之前評估預測。

先決條件

若要在預測擴展政策中新增自訂指標，您必須擁有 `cloudwatch:GetMetricData` 許可。

若要指定自己的指標，而不是 AWS 提供的指標，您必須先將指標發佈至 CloudWatch。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[發佈自訂指標](#)。

如果您發佈自己的指標，應確保以至少五分鐘的頻率發佈資料點。Amazon EC2 Auto Scaling 根據需要的時段長度從 CloudWatch 擷取資料點。例如，負載指標規範使用每小時指標來衡量應用程式的負載。CloudWatch 使用您發佈的指標資料，透過將時間戳記落於同一小時週期內的所有資料點彙總，為任何一小時週期提供單一資料值。

建構自訂指標的 JSON

下一節包含如何設定預測擴展以查詢來自 CloudWatch 的資料的範例。設定此選項有兩種不同的方法，而您選擇的方法會影響您用來建構預測擴展政策的 JSON 的格式。使用指標數學時，JSON 的格式會根據所執行的指標數學而進一步變化。

1. 若要建立直接從 AWS 您發佈至 CloudWatch 的其他 CloudWatch 指標取得資料的政策，請參閱 [具有自訂負載和擴展指標的預測擴展政策範例 \(AWS CLI\)](#)。
2. 若要建立能查詢多個 CloudWatch 指標並使用數學表達式根據這些指標來建立新的時間序列的政策，請參閱 [使用指標數學表達式](#)。

具有自訂負載和擴展指標的預測擴展政策範例 (AWS CLI)

若要使用 建立具有自訂載入和擴展指標的預測擴展政策 AWS CLI，請將 的引數存放在名為 `--predictive-scaling-configuration` 的 JSON 檔案中 `config.json`。

您可以藉由將以下範例中的可替換值換成您的指標和目標使用率的值，開始新增自訂指標。

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
```

```

        "Namespace": "MyNameSpace",
        "Dimensions": [
            {
                "Name": "MyOptionalMetricDimensionName",
                "Value": "MyOptionalMetricDimensionValue"
            }
        ]
    },
    "Stat": "Average"
}
]
},
"CustomizedLoadMetricSpecification": {
    "MetricDataQueries": [
        {
            "Id": "load_metric",
            "MetricStat": {
                "Metric": {
                    "MetricName": "MyLoadMetric",
                    "Namespace": "MyNameSpace",
                    "Dimensions": [
                        {
                            "Name": "MyOptionalMetricDimensionName",
                            "Value": "MyOptionalMetricDimensionValue"
                        }
                    ]
                },
                "Stat": "Sum"
            }
        }
    ]
}
}
]
}
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [MetricDataQuery](#)。

Note

以下是一些其他資源，可協助您尋找 CloudWatch 指標的指標名稱、命名空間、維度和統計資料：

- 如需 AWS 服務可用指標的相關資訊，請參閱《Amazon [AWS CloudWatch 使用者指南](#)》中的 [發佈 CloudWatch 指標](#) 的服務。Amazon CloudWatch
- 若要使用取得 CloudWatch 指標的確切指標名稱、命名空間和維度（如適用）AWS CLI，請參閱 [清單指標](#)。

若要建立此政策，執行使用 JSON 檔案作為輸入 [put-scaling-policy](#) 命令，如以下範例所示。

```
aws autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \  
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令會傳回政策的 Amazon Resource Name (ARN)。

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",  
  "Alarms": []  
}
```

使用指標數學表達式

下一節提供預測擴展政策的資訊和範例，說明如何在政策中使用指標數學。

主題

- [了解指標數學](#)
- [使用指標數學組合指標的預測擴展政策範例 \(AWS CLI\)](#)
- [藍/綠部署情境中使用的預測擴展政策範例 \(AWS CLI\)](#)

了解指標數學

如果您只想彙總現有指標資料，則 CloudWatch 指標數學可以節省將另一個指標發佈至 CloudWatch 的工作量和成本。您可以使用 AWS 提供的任何指標，也可以使用您在應用程式中定義的指標。例如，您可能會想要計算每個執行個體的 Amazon SQS 佇列待處理項目。您可以透過從佇列中取得可檢索的大致訊息數量，然後將該數字除以 Auto Scaling 群組的執行中容量來得出此結果。

如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [使用指標數學](#)。

如果選擇在預測擴展政策中使用指標數學表達式，則請考慮以下幾點：

- 指標數學運算會使用指標名稱、命名空間和指標維度鍵/值對之唯一組合的資料點。
- 您可以使用任何算術運算子 (+-*/^)、統計函數 (如 AVG 或 SUM) 或 CloudWatch 支援的其他函數。
- 您可以在數學表達式的公式中同時使用其他數學表達式的指標和結果。
- 您的指標數學表達式可以由不同的彙總組成。但是，最終彙總結果的最佳實務是將 Average 用於擴展指標，Sum 用於負載指標。
- 在指標規範中使用的任何表達式都必須最終傳回單一的時間序列。

若要使用指標數學，請執行以下操作：

- 選擇一或多個 CloudWatch 指標。然後，建立表達式。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[使用指標數學](#)。
- 透過使用 CloudWatch 主控台或 CloudWatch [GetMetricData](#) API 驗證指標數學表達式是否有效。

使用指標數學組合指標的預測擴展政策範例 (AWS CLI)

有時，您可能需要首先以某種方式處理其資料，而不是直接指定指標。例如，您可能有一個從 Amazon SQS 佇列中提取工作的應用程式，並且您可能想要使用佇列中的項目數量作為預測擴展的條件。佇列中的訊息數量並不僅僅定義所需的執行個體數量。因此，需要更多的工作來建立可用於計算每個執行個體待處理項目的指標。如需詳細資訊，請參閱[以 Amazon SQS 為基礎的擴展政策](#)。

以下是此案例的預測擴展政策範例。它指定了基於 Amazon SQS ApproximateNumberOfMessagesVisible 指標的擴展和負載指標，即可從佇列中擷取的訊息數量。它還使用 Amazon EC2 Auto Scaling GroupInServiceInstances 指標和數學表達式來計算每個執行個體的待處理項目，以擴展指標。

```
aws autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \  
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \  
  --predictive-scaling-configuration file://config.json  
{  
  "MetricSpecifications": [  
    {  
      "TargetValue": 100,  
      "CustomizedScalingMetricSpecification": {  
        "MetricDataQueries": [  
          {
```

```

        "Label": "Get the queue size (the number of messages waiting to be
processed)",
        "Id": "queue_size",
        "MetricStat": {
            "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                    {
                        "Name": "QueueName",
                        "Value": "my-queue"
                    }
                ]
            },
            "Stat": "Sum"
        },
        "ReturnData": false
    },
    {
        "Label": "Get the group size (the number of running instances)",
        "Id": "running_capacity",
        "MetricStat": {
            "Metric": {
                "MetricName": "GroupInServiceInstances",
                "Namespace": "AWS/AutoScaling",
                "Dimensions": [
                    {
                        "Name": "AutoScalingGroupName",
                        "Value": "my-asg"
                    }
                ]
            },
            "Stat": "Sum"
        },
        "ReturnData": false
    },
    {
        "Label": "Calculate the backlog per instance",
        "Id": "scaling_metric",
        "Expression": "queue_size / running_capacity",
        "ReturnData": true
    }
]
},

```

```

    "CustomizedLoadMetricSpecification": {
      "MetricDataQueries": [
        {
          "Id": "load_metric",
          "MetricStat": {
            "Metric": {
              "MetricName": "ApproximateNumberOfMessagesVisible",
              "Namespace": "AWS/SQS",
              "Dimensions": [
                {
                  "Name": "QueueName",
                  "Value": "my-queue"
                }
              ],
            },
            "Stat": "Sum"
          },
          "ReturnData": true
        }
      ]
    }
  ]
}

```

該範例會傳回政策的 ARN。

```

{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
  "Alarms": []
}

```

藍/綠部署情境中使用的預測擴展政策範例 (AWS CLI)

搜尋表達式提供了進階選項，您可以在該選項中查詢多個 Auto Scaling 群組中的指標，並對其執行數學表達式。這對藍/綠部署尤其實用。

Note

藍/綠部署是一種部署方法，您可以在其中建立兩個獨立但相同的 Auto Scaling 群組。只有一個群組會接收生產流量。使用者流量最初定向至較早的 Auto Scaling 群組（「藍色」），而新群組

(「綠色」) 用於測試和評估應用程式或服務的新版本。測試並接受新部署後，使用者流量將轉移至綠色 Auto Scaling 群組。然後，您可以在部署成功後刪除藍色群組。

當新 Auto Scaling 群組作為藍/綠色部署的一部分建立時，可以自動將每個群組的指標歷史記錄包含在預測擴展政策中，而無需變更其指標規範。如需詳細資訊，請參閱 AWS 運算部落格上的[將 EC2 Auto Scaling 預測擴展政策與藍/綠部署搭配使用](#)。

以下範例政策示範如何執行此操作。在此範例中，政策使用 Amazon EC2 發出的 CPUUtilization 指標。它使用 Amazon EC2 Auto Scaling GroupInServiceInstances 指標和數學表達式來計算每個執行個體的擴展指標值。它還指定了容量指標規範以取得 GroupInServiceInstances 指標。

搜尋表達式會根據指定的搜尋條件尋找多個 Auto Scaling 群組中執行個體的 CPUUtilization。如果您稍後建立了與相同搜尋條件相符的新 Auto Scaling 群組，則會自動包含新 Auto Scaling 群組中執行個體的 CPUUtilization。

```
aws autoscaling put-scaling-policy --policy-name my-blue-green-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 25,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=\"CPUUtilization\" ASG-myapp', 'Sum', 300))",
            "ReturnData": false
          },
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName} MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))",
            "ReturnData": false
          },
          {
            "Id": "weighted_average",
            "Expression": "load_sum / capacity_sum",
            "ReturnData": true
          }
        ]
      }
    }
  ]
}
```

```

    }
  ]
},
"CustomizedLoadMetricSpecification": {
  "MetricDataQueries": [
    {
      "Id": "load_sum",
      "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=\\\"CPUUtilization\\\" ASG-myapp', 'Sum', 3600))"
    }
  ]
},
"CustomizedCapacityMetricSpecification": {
  "MetricDataQueries": [
    {
      "Id": "capacity_sum",
      "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName} MetricName=\\\"GroupInServiceInstances\\\" ASG-myapp', 'Average', 300))"
    }
  ]
}
}
]
}

```

該範例會傳回政策的 ARN。

```

{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-blue-green-predictive-scaling-policy",
  "Alarms": []
}

```

預測擴展政策中自訂指標的考量事項

如果使用自訂指標時出現問題，建議您執行以下操作：

- 如果提供了錯誤訊息，則請閱讀該訊息，並在可行的狀況下解決其報告的問題。
- 如果嘗試在藍色/綠色部署案例中使用搜尋表達式時出現問題，則請首先確保您了解如何建立查找部分相符而不是完全相符的搜尋表達式。此外，請檢查查詢是否僅尋找執行特定應用程式的 Auto

Scaling 群組。如需搜尋表達式語法的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [CloudWatch 搜尋表達式語法](#)。

- 如果未事先驗證表達式，則 [put-scaling-policy](#) 命令會在建立擴展政策時對其進行驗證。但是，此命令可能無法識別偵測到之錯誤的確切原因。若要修正這些問題，請對您收到之 [get-metric-data](#) 命令請求回應中的錯誤進行疑難排解。您也可以從 CloudWatch 主控台對表達式進行疑難排解。
- 當您在主控台中檢視 Load (負載) 和 Capacity (容量) 圖表時，Capacity (容量) 圖表可能不會顯示任何資料。要確保圖表具有完整的資料，請確保始終如一地為 Auto Scaling 群組啟用群組指標。如需詳細資訊，請參閱[啟用 Auto Scaling 群組指標 \(主控台\)](#)。
- 當應用程式在其生命週期內於不同的 Auto Scaling 群組中執行時，容量指標規範僅對藍/綠色部署有用。此自訂指標允許您提供多個 Auto Scaling 群組的總容量。預測擴展使用此選項來顯示主控台 Capacity (容量) 圖表中的歷史資料。
- 如果 MetricDataQueries 自己指定了 SEARCH() 函數 (在無需 SUM() 等數學函數的狀況下)，則您必須為 ReturnData 指定 false。這是因為搜尋表達式可能會傳回多個時間序列，並且基於表達式的指標規範只能傳回一個時間序列。
- 搜尋表達式中涉及的所有指標均應具有相同的解析度。

限制

- 您可以在一個指標規範中查詢最多 10 個指標的資料點。
- 在此限制之下，一個表達式計為一個指標。

控制縮減期間應終止的 Auto Scaling 執行個體

Amazon EC2 Auto Scaling 使用終止政策來決定終止執行個體的順序。您可以使用預先定義的政策或建立自訂政策來滿足您的特定需求。透過使用自訂政策或執行個體縮減保護，您也可以防止 Auto Scaling 群組終止尚未準備好終止的執行個體。

目錄

- [當 Amazon EC2 Auto Scaling 使用終止政策時](#)
- [設定 Amazon EC2 Auto Scaling 的終止政策](#)
- [使用 Lambda 建立自訂終止政策](#)
- [使用執行個體縮減保護來控制執行個體終止](#)
- [設計您的應用程式以正常處理執行個體終止](#)

當 Amazon EC2 Auto Scaling 使用終止政策時

以下各節說明 Amazon EC2 Auto Scaling 使用終止政策的情境。

目錄

- [在事件中擴展](#)
- [執行個體重新整理](#)
- [可用區域重新平衡](#)

在事件中擴展

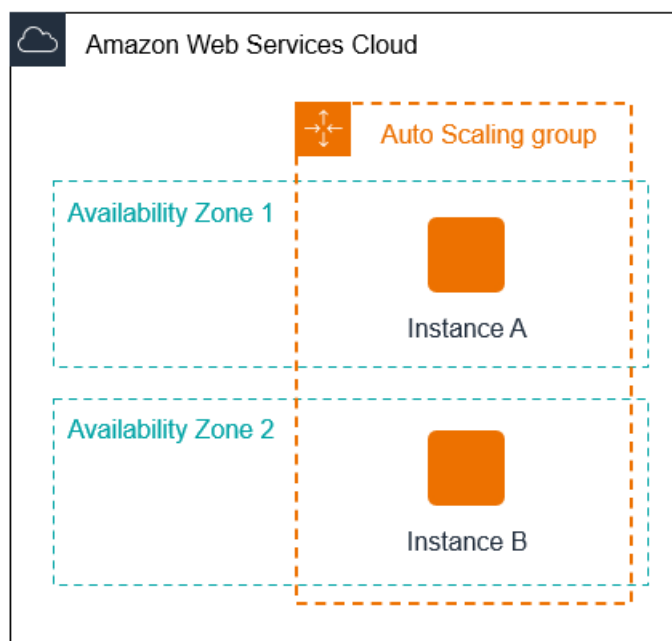
當 Auto Scaling 群組所需容量的新值低於群組的目前容量時，就會發生擴展事件。

事件的擴展會在下列案例中發生：

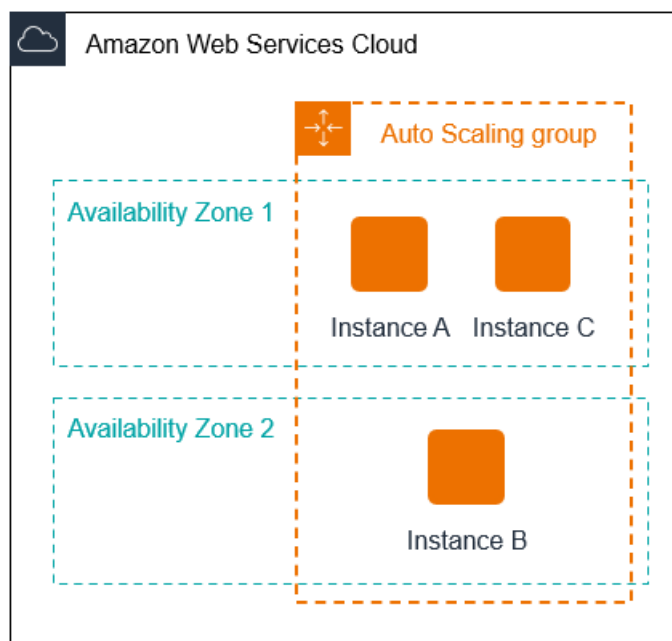
- 使用動態擴展政策時，群組的大小會因指標值變更而減少
- 使用排程擴展時，群組的大小會因排程動作而減少
- 手動縮減群組的大小

下列範例顯示發生擴展時終止政策的運作方式。

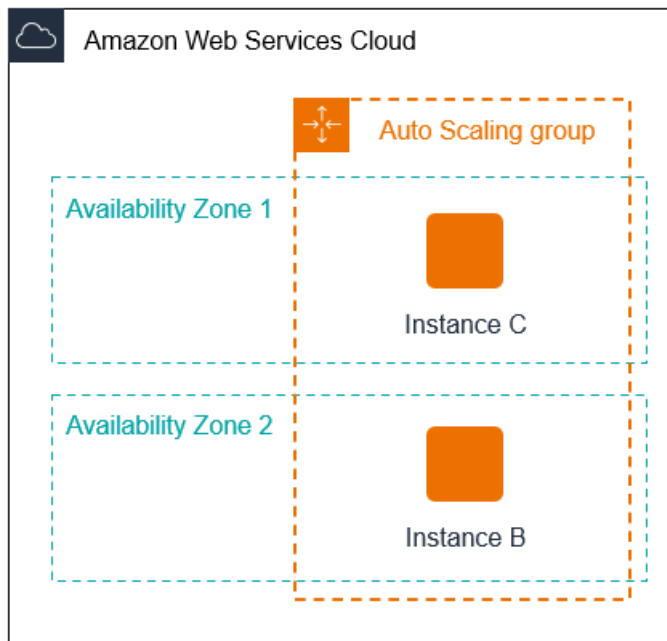
1. 本例中的 Auto Scaling 群組具有一個執行個體類型、兩個可用區域以及兩個執行個體所需容量。它也具有動態擴展政策，可在資源使用率增加或減少時新增和移除執行個體。此群組中的執行個體會分佈在兩個可用區域，如下圖所示。



2. 當 Auto Scaling 群組擴增時，Amazon EC2 Auto Scaling 會啟動新的執行個體。Auto Scaling 群組現在有三個執行個體，分佈在兩個可用區域，如下圖所示。



3. 當 Auto Scaling 群組縮減時，Amazon EC2 Auto Scaling 會終止其中一個執行個體。
4. 如果您並沒有為群組指定終止政策，則 Amazon EC2 Auto Scaling 會採用預設的終止政策。它會選取具有兩個執行個體的可用區域，並終止從啟動組態、不同啟動範本或目前啟動範本最舊版本的執行個體。如果執行個體是從相同的啟動範本和版本啟動，Amazon EC2 Auto Scaling 會選取最接近下一個計費小時的執行個體，並將其終止。



執行個體重新整理

您可以啟動執行個體重新整理，以更新 Auto Scaling 群組中的執行個體。在執行個體重新整理期間，Amazon EC2 Auto Scaling 會終止群組中的執行個體，然後啟動已終止執行個體的取代項目。Auto Scaling 群組的終止政策可控制要優先取代哪些執行個體。

可用區域重新平衡

Amazon EC2 Auto Scaling 可在為 Auto Scaling 群組啟用的可用區域之間平均平衡容量。這有助於減少可用區域中斷所造成的影響。如果可用區域之間的容量分佈不平衡，Amazon EC2 Auto Scaling 會透過在具有最少執行個體的已啟用可用區域中啟動執行個體，並在其他地方終止執行個體，來重新平衡 Auto Scaling 群組。終止政策會控制要優先終止哪些執行個體。

有許多可能的原因會造成可用區域之間的執行個體分佈失去平衡。

移除執行個體

如果您從 Auto Scaling 群組分開執行個體、將執行個體置於待命狀態，或明確終止執行個體並減少所需的容量，從而防止取代執行個體啟動，群組可能會呈現不平衡狀態。若發生此情況，Amazon EC2 Auto Scaling 會重新平衡可用區域。

使用不同於原始指定的可用區域

如果您擴展 Auto Scaling 群組以包含其他可用區域，或變更使用的可用區域，則 Amazon EC2 Auto Scaling 會啟動新可用區域中的執行個體，並終止其他區域中的執行個體，以協助確保 Auto Scaling 群組會平均分佈於可用區域。

可用性中斷

可用性中斷並不常見。不過，如果某一個可用區域變成無法使用的狀態但稍後復原，則 Auto Scaling 群組會在可用區域間呈現不平衡狀態。Amazon EC2 Auto Scaling 會嘗試逐步重新平衡群組，而重新平衡可能會終止其他區域中的執行個體。

例如，您的 Auto Scaling 群組具有一個執行個體類型、兩個可用區域以及兩個執行個體所需容量。在一個可用區域發生故障的情況下，Amazon EC2 Auto Scaling 會自動啟動運作狀態良好的可用區域中的新執行個體，以取代運作狀態不佳的可用區域中的執行個體。然後，在運作狀態不佳的可用區域稍後恢復正常狀態時，Amazon EC2 Auto Scaling 會自動啟動此區域中的新執行個體，進而終止未受影響區域中的執行個體。

Note

當重新平衡時，Amazon EC2 Auto Scaling 會在終止舊的應用程式執行個體前啟動一個新的執行個體。如此一來，重新平衡並不會影響效能或應用程式的可用性。

由於 Amazon EC2 Auto Scaling 會在終止舊的執行個體前，嘗試啟動新執行個體，當使用或接近最大指定容量時，會阻礙或完全停止重新平衡的活動。若要避免這個問題，系統在重新平衡活動時，可以暫時使用超過群組所指定容量上限的 10% (或可使用超過一個執行個體的容量，以容量較大者優先)。僅在群組位於或接近容量上限且需要重新平衡時，容量的邊界會延伸，因為使用者請求重新分區，或補償區域可用性問題。只要有需要重新平衡群組，邊界就會延伸。

設定 Amazon EC2 Auto Scaling 的終止政策

終止政策提供 Amazon EC2 Auto Scaling 依特定順序終止執行個體所遵循的條件。根據預設，Amazon EC2 Auto Scaling 會使用終止政策，其設計旨在先終止使用過時組態的執行個體。您可以變更終止政策，以控制哪些執行個體最需要先終止。

當 Amazon EC2 Auto Scaling 終止執行個體時，它會嘗試在為 Auto Scaling 群組啟用的可用區域之間維持平衡。維持區域平衡優先於終止政策。如果一個可用區域具有比其他可用區域更多的執行個體，Amazon EC2 Auto Scaling 會先將終止政策套用至不平衡的區域。如果可用區域是平衡的，則會在所有區域中套用終止政策。

主題

- [預設終止政策的運作方式](#)
- [預設終止政策和混合執行個體群組](#)
- [預先定義的終止政策](#)
- [變更 Auto Scaling 群組的終止政策](#)

Note

Amazon EC2 Auto Scaling 只會將終止政策套用至 Auto Scaling 群組認為運作狀態不佳的執行個體。因此，Auto Scaling 運作狀態檢查標示為運作狀態不佳的執行個體將略過終止政策評估。

如需詳細資訊，請參閱[設計您的應用程式以正常處理執行個體終止](#)。

預設終止政策的運作方式

當 Amazon EC2 Auto Scaling 需要終止執行個體時，它會先識別哪個可用區域（或區域）具有最多的執行個體，以及至少一個不受縮減保護的執行個體。然後，它會繼續評估已識別可用區域內未受保護的執行個體，如下所示：

使用過時組態的執行個體

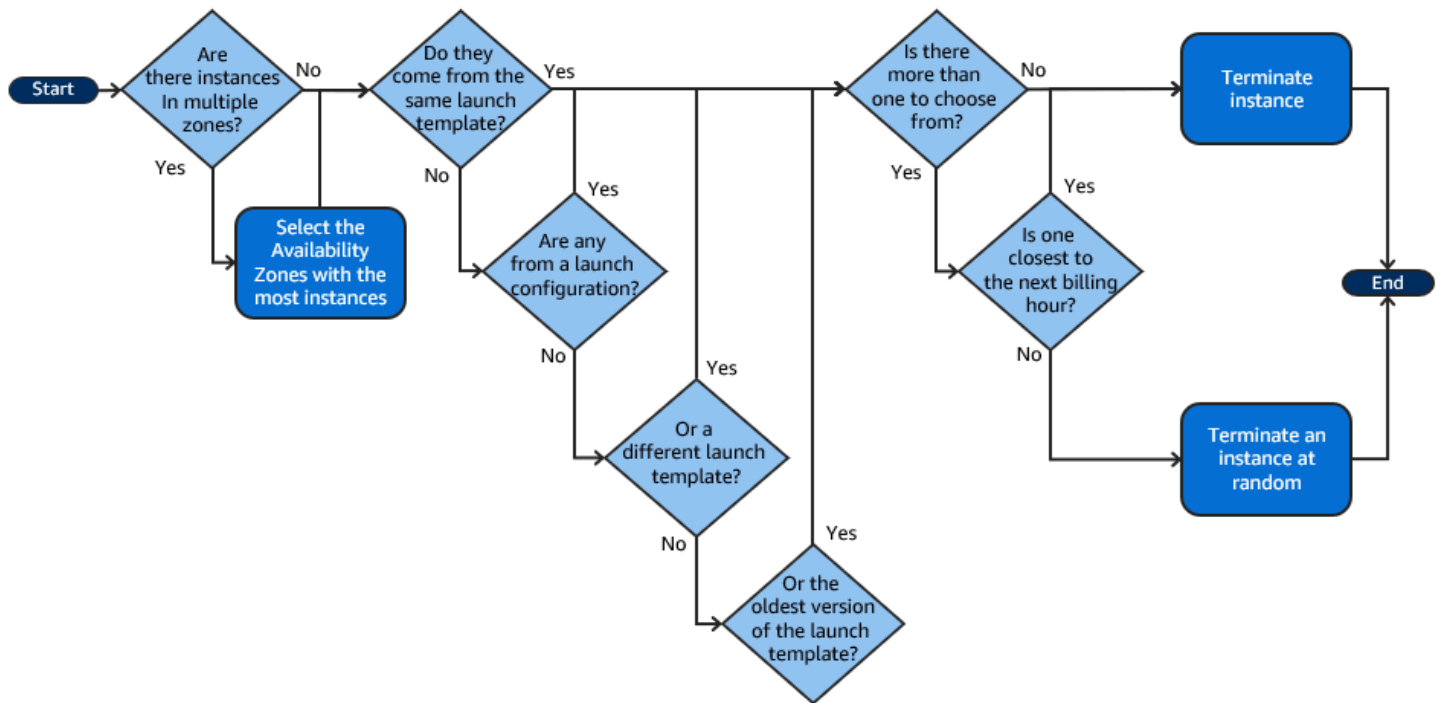
- 對於使用啟動範本的群組 – 判斷是否有任何執行個體使用過時的組態，依此順序排定優先順序：
 1. 首先，檢查使用啟動組態啟動的執行個體。
 2. 然後，檢查使用不同啟動範本而非目前啟動範本啟動的執行個體。
 3. 最後，使用目前啟動範本的最新版本來檢查執行個體。
- 對於使用啟動組態的群組 – 判斷是否有任何執行個體使用最舊的啟動組態。

如果找不到具有過時組態的執行個體，或者有多個執行個體可供選擇，Amazon EC2 Auto Scaling 會考慮下一個接近下一個計費小時的執行個體條件。

接近下一個計費小時的執行個體

判斷符合先前條件的任何執行個體是否最接近下一個計費小時。如果多個執行個體同樣關閉，請隨機終止一個執行個體。這可協助您最大限度地使用每小時計費的執行個體。不過，大多數 EC2 用量現在是每秒計費，因此此最佳化可提供較少的好處。如需詳細資訊，請參閱[Amazon EC2 定價](#)。

下列流程圖說明預設終止政策對使用啟動範本的群組的運作方式。



預設終止政策和混合執行個體群組

在[混合執行個體群組](#)中終止執行個體時，Amazon EC2 Auto Scaling 會套用其他條件。

當 Amazon EC2 Auto Scaling 需要終止執行個體時，它首先會識別應該根據群組的設定終止哪個購買選項 (Spot 或隨需)。這可確保群組隨著時間推移趨向 Spot 和隨需執行個體的指定比率。

然後，它會在每個可用區域內獨立套用終止政策。它會決定要在哪個 Spot 或隨需執行個體中終止可用區域，以保持可用區域平衡。相同的邏輯適用於具有為執行個體類型定義權重的混合執行個體群組。

在每個區域中，預設終止政策的運作方式如下，以判斷識別的購買選項中可以終止哪些未受保護的執行個體：

1. 判斷是否可以終止任何執行個體，以改善與 Auto Scaling 群組指定[配置策略](#)的一致性。如果未識別要最佳化的執行個體，或者有多個執行個體可供選擇，則評估會繼續進行。
2. 判斷是否有任何執行個體使用過時的組態，並依此順序排定優先順序：
 - a. 首先，檢查使用啟動組態啟動的執行個體。
 - b. 然後，檢查使用不同啟動範本而非目前啟動範本啟動的執行個體。
 - c. 最後，使用目前啟動範本的最新版本來檢查執行個體。

如果找不到具有過時組態的執行個體，或者有多個執行個體可供選擇，則評估會繼續進行。

3. 判斷是否有任何執行個體最接近下一個計費小時。如果多個執行個體同樣接近，請隨機選擇一個執行個體。

預先定義的終止政策

您可以從下列預先定義的終止政策中選擇：

- **Default** – 根據預設終止政策終止執行個體。
- **AllocationStrategy** – 終止 Auto Scaling 群組中的執行個體，讓剩餘的執行個體符合正在終止之執行個體類型的配置策略 (Spot 執行個體或隨需執行個體)。當您偏好使用的執行個體類型變更時，此政策非常有用。如果 Spot 配置策略是 `lowest-price`，您可以逐步重新平衡 Spot 執行個體在 N 個最低價 Spot 集區中的分佈。如果 Spot 配置策略是 `capacity-optimized`，您可以逐步重新平衡 Spot 執行個體在有更多可用 Spot 容量的 Spot 集區中的分佈。您也可以使用優先順序較高類型的隨需執行個體逐步取代優先順序較低類型的隨需執行個體。
- **OldestLaunchTemplate** – 終止具有最舊啟動範本的執行個體。使用此政策時，使用非最新啟動範本的執行個體會先被終止，然後是使用目前啟動範本最舊版本的執行個體。當您更新群組，並從先前的組態中淘汰執行個體時，此政策非常有用。
- **OldestLaunchConfiguration** – 終止具有最舊啟動組態的執行個體。當您更新群組，並從先前的組態中淘汰執行個體時，此政策非常有用。使用此政策時，使用非最新啟動組態的執行個體會先被終止。
- **ClosestToNextInstanceHour** – 終止最接近下一個計費小時的執行個體。此政策有助於充分利用您每小時計費的執行個體。
- **NewestInstance** – 終止群組中最新的執行個體。此政策非常實用，當您測試新的啟動組態時，但又不想讓在生產環境中。
- **OldestInstance** – 終止群組中最舊的執行個體。當您將 Auto Scaling 群組中的執行個體升級為新的 EC2 執行個體類型時，此選項非常實用。您可以逐步取代舊類型的執行個體，將它們換成新的類型。

Note

不論使用何種終止政策，Amazon EC2 Auto Scaling 一律會先平衡可用區域之間的執行個體。因此，您可能會遇到某些較新的執行個體在較舊的執行個體之前終止的情況。例如，最近有新增的可用區域，或是某個可用區域的執行個體比另一個由群組使用的可用區域還要多等情況。

變更 Auto Scaling 群組的終止政策

若要變更 Auto Scaling 群組的終止政策，請使用下列其中一種方法。

Console

當您最初在 Amazon EC2 Auto Scaling 主控台中建立 Auto Scaling 群組時，您無法變更終止政策。系統將會自動使用預設終止政策。建立 Auto Scaling 群組之後，您可以將預設政策取代為不同的終止政策，或依應該套用的順序列出的多個終止政策。

變更 Auto Scaling 群組的終止政策

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Advanced configurations (進階組態)、Edit (編輯)。
4. 對於 Termination policies (終止政策)，請選擇一或多個終止政策。如果選擇多個政策，則依想要對其進行評估的順序排列。

您可以選擇性地選擇 Custom termination policy (自訂終止政策)，然後選擇符合您需求的 Lambda 函數。如果您已經為 Lambda 函式建立了版本和別名，則可以從 Version/Alias (版本/別名) 下拉式選單中選擇版本或別名。若要使用 Lambda 函數的未發佈版本，請保留 Version/Alias (版本/別名) 設定為預設值。如需詳細資訊，請參閱[使用 Lambda 建立自訂終止政策](#)。

Note

使用多個政策時，必須正確設定其順序：

- 如果您使用 Default (預設) 政策，則其必須是清單的最後一項政策。
- 如果您使用 Custom termination policy (自訂終止政策)，則其必須是清單中的第一項政策。

5. 選擇更新。

AWS CLI

除非指定了不同的政策，否則系統會自動使用預設終止政策。

變更 Auto Scaling 群組的終止政策

請使用以下其中一個命令：

- [create-auto-scaling-group](#)
- [update-auto-scaling-group](#)

您可以分別單獨使用終止政策，或將其合併成政策清單。例如，利用下列的命令來更新 Auto Scaling 群組，以先使用 OldestLaunchConfiguration 政策，然後再使用 ClosestToNextInstanceHour 政策。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --  
termination-policies "OldestLaunchConfiguration" "ClosestToNextInstanceHour"
```

如果您使用 Default 終止政策，使其列於終止政策清單的最後一項。例如 --termination-policies "OldestLaunchConfiguration" "Default"。

若要使用自訂終止政策，您必須先使用 建立終止政策 AWS Lambda。若要指定 Lambda 函數作為終止政策，請將其設定為終止政策清單中的第一項。例如 --termination-policies "arn:aws:lambda:us-west-2:123456789012:function:HelloFunction:prod" "OldestLaunchConfiguration"。如需詳細資訊，請參閱[使用 Lambda 建立自訂終止政策](#)。

使用 Lambda 建立自訂終止政策

Amazon EC2 Auto Scaling 使用終止政策來排定優先順序，以確定在減少 Auto Scaling 群組的大小 (稱為縮減) 時要先終止哪些執行個體。Auto Scaling 群組會使用預設的終止政策，但您可以選擇性地選擇或建立自己的終止政策。如需有關如何選擇預先定義的終止政策的詳細資訊，請參閱[設定 Amazon EC2 Auto Scaling 的終止政策](#)。

在本主題中，您會了解如何使用 Amazon EC2 Auto Scaling 為回應特定事件而叫用的 AWS Lambda 函數建立自訂終止政策。您建立的 Lambda 函數會處理 Amazon EC2 Auto Scaling 所傳送的輸入資料中的資訊，並傳回準備終止的執行個體清單。

自訂終止政策可針對哪要終止些執行個體及何時終止提供更好的控制。例如，當您的 Auto Scaling 群組縮減時，Amazon EC2 Auto Scaling 無法確定是否有不應中斷的工作負載正在執行。使用 Lambda 函數，您可以驗證終止請求，並等到工作負載完成後，再將執行個體 ID 傳回 Amazon EC2 Auto Scaling 以進行終止。

目錄

- [輸入資料](#)
- [回應資料](#)
- [考量事項](#)
- [建立 Lambda 函式](#)
- [限制](#)

輸入資料

Amazon EC2 Auto Scaling 會為縮減事件產生 JSON 承載，並在執行個體因執行個體生命週期上限或執行個體重新整理功能而即將終止時執行此操作。它也會為擴展事件產生 JSON 承載，以便在跨可用區域重新平衡群組時啟動。

此承載包含有關 Amazon EC2 Auto Scaling 需要終止的容量、建議終止的執行個體清單，以及啟動終止的事件等資訊。

以下是承載範例：

```
{
  "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:<account-id>:autoScalingGroup:d4738357-2d40-4038-ae7e-b00ae0227003:autoScalingGroupName/my-asg",
  "AutoScalingGroupName": "my-asg",
  "CapacityToTerminate": [
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 2,
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 1,
      "InstanceMarketOption": "spot"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "Capacity": 3,
      "InstanceMarketOption": "on-demand"
    }
  ],
  "Instances": [
```

```
{
  "AvailabilityZone": "us-east-1b",
  "InstanceId": "i-0056faf8da3e1f75d",
  "InstanceType": "t2.nano",
  "InstanceMarketOption": "on-demand"
},
{
  "AvailabilityZone": "us-east-1c",
  "InstanceId": "i-02e1c69383a3ed501",
  "InstanceType": "t2.nano",
  "InstanceMarketOption": "on-demand"
},
{
  "AvailabilityZone": "us-east-1c",
  "InstanceId": "i-036bc44b6092c01c7",
  "InstanceType": "t2.nano",
  "InstanceMarketOption": "on-demand"
},
...
],
"Cause": "SCALE_IN"
}
```

承載資源包括 Auto Scaling 群組的名稱、其 Amazon Resource Name (ARN)，以及下列元素：

- CapacityToTerminate 描述您的 Spot 或隨需容量中有多少數量設定為在指定可用區域中終止。
- Instances 代表 Amazon EC2 Auto Scaling 根據 CapacityToTerminate 中的資訊而建議終止的執行個體。
- Cause 描述引發終止的事件：SCALE_IN、INSTANCE_REFRESH、MAX_INSTANCE_LIFETIME 或 REBALANCE。

以下資訊概述了 Amazon EC2 Auto Scaling 如何在輸入資料中產生 Instances 的最重要因素：

- 當執行個體因擴展事件和執行個體重新整理型終止而終止時，會優先在可用區域之間維持平衡。因此，如果某一個可用區域的執行個體比另一個由群組使用的可用區域還要多，則輸入資料會包含僅來自不平衡可用區域且符合終止資格的執行個體。如果群組使用的可用區域處於平衡狀態，則輸入資料會包含來自群組中的所有可用區域的執行個體。
- 使用[混合執行個體政策](#)時，根據每個購買選項所需的百分比，維持 Spot 和隨需容量的平衡也具有優先權。我們首先要識別應終止兩種類型 (Spot 或隨需) 中的哪一種。接著，我們會識別我們可以在哪些可用區域中終止哪些執行個體 (在已識別的購買選項內) 將會使可用區域最為平衡。

回應資料

輸入資料和回應資料一起運作，以縮減要終止的執行個體清單。

使用給定的輸入，Lambda 函數的回應應類似於下列範例：

```
{
  "InstanceIDs": [
    "i-02e1c69383a3ed501",
    "i-036bc44b6092c01c7",
    ...
  ]
}
```

回應中的 InstanceIDs 代表準備終止的執行個體。

您也可以傳回準備終止的一組不同執行個體，這會覆寫輸入資料中的執行個體。如果在叫用 Lambda 函數時沒有準備好終止的執行個體，您也可以選擇不傳回任何執行個體。

沒有執行個體可以終止時，Lambda 函數的回應應類似於以下範例：

```
{
  "InstanceIDs": [ ]
}
```

考量事項

使用自訂終止政策時，請注意以下注意事項：

- 先在回應資料中傳回執行個體並不能保證其會終止。如果在叫用 Lambda 函數時傳回的執行個體數量超過所需數量，Amazon EC2 Auto Scaling 會根據您為 Auto Scaling 群組指定的其他終止政策來評估每個執行個體。當有多個終止政策時，會嘗試套用清單中的下一個終止政策，而且如果執行個體數量多於需終止的數量，則會繼續執行下一個終止政策，依此類推。如果未指定其他終止政策，則會使用預設終止政策來判斷要終止哪些執行個體。
- 如果未傳回執行個體或 Lambda 函數逾時，Amazon EC2 Auto Scaling 會等待一小段時間，然後再次叫用您的函數。對於任何規模事件，只要群組所需的容量小於其目前的容量，就會繼續嘗試。對於執行個體重新整理型終止，則會繼續嘗試一個小時。之後，如果仍然無法終止任何執行個體，執行個體重新整理作業即會失敗。透過執行個體生命週期上限設定，Amazon EC2 Auto Scaling 會繼續嘗試終止被識別為超過其生命週期上限的執行個體。

- 因為函數會重複重試，所以請務必在使用 Lambda 函數作為自訂終止政策之前，先測試並修正程式碼中的任何永久錯誤。
- 如果您使用自己的待終止執行個體清單來覆寫輸入資料，而終止這些執行個體會使可用區域失去平衡，則 Amazon EC2 Auto Scaling 會逐步重新平衡可用區域之間的容量分佈。首先，系統會叫用 Lambda 函數來查看是否有準備終止的執行個體，以確定是否要開始重新平衡。如果有執行個體準備要終止，則會先啟動新的執行個體。當執行個體完成啟動時，它會偵測到您群組的目前容量高於其所需的容量，並在事件發生時啟動擴展。
- 自訂終止政策不會影響您也使用擴展保護來保護特定執行個體免於終止的能力。如需詳細資訊，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。
- 如果您的 Lambda 函數未傳回任何執行個體，則不會自動防止所有終止。無論終止政策為何，Amazon EC2 Auto Scaling 仍會終止 Auto Scaling 運作狀態檢查視為運作狀態不佳的執行個體。

建立 Lambda 函式

首先會建立 Lambda 函數，以便您可以在 Auto Scaling 群組的終止政策中指定其 Amazon Resource Name (ARN)。

建立 Lambda 函數 (主控台)

1. 開啟 Lambda 主控台中的 [Functions \(函數\) 頁面](#)。
2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所用的相同區域。
3. 依序選擇 Create function (建立函數)、Author from scratch (從頭開始撰寫)。
4. 在 Basic information (基本資訊) 下，對於 Function name (函數名稱)，為您的函數輸入名稱。
5. 選擇 Create function (建立函數)。您會回到該函數的程式碼和組態畫面。
6. 在函數仍在主控台中開啟的情況下，請在 Function code (函數程式碼) 下，將程式碼貼入編輯器。
7. 選擇部署。
8. 也可以透過選擇 Versions (版本) 索引標籤來建立 Lambda 函數的已發佈版本，然後 Publish new version (發佈新版本)。若要進一步了解 Lambda 中的版本控制，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 函數版本](#)。
9. 如果選擇發佈版本，並且要將別名與此版本的 Lambda 函數建立關聯，請選擇 Aliases (別名) 索引標籤。若要進一步了解 Lambda 中的別名，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 函數別名](#)。
10. 接著依序選擇 Configuration (組態) 索引標籤、Permissions (許可)。

11. 向下捲動至 Resource-based policy (資源型政策)，然後選擇 Add permissions (新增許可)。資源型政策可用於向政策中所指定的委託人授予叫用函數的許可。在這種情況下，委託人是與 Auto Scaling 群組相關聯的 [Amazon EC2 Auto Scaling 服務連結角色](#)。
12. 在 Policy statement (政策陳述式) 區段中，設定您的許可：
 - a. 選擇 AWS 帳戶。
 - b. 在 Principal (委託人) 中，輸入呼叫服務連結角色的 ARN，例如 **arn:aws:iam::<aws-account-id>:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling**。
 - c. 在 Action (動作) 中，選擇 **lambda:InvokeFunction**。
 - d. 在 Statement ID (陳述式 ID) 中，請輸入唯一的陳述式 ID，例如 **AllowInvokeByAutoScaling**。
 - e. 選擇儲存。
13. 遵照這些指示後，請繼續進行下一步，在 Auto Scaling 群組的終止政策中指定函數的 ARN。如需詳細資訊，請參閱[變更 Auto Scaling 群組的終止政策](#)。

Note

如需可用作開發 Lambda 函數參考的範例，請參閱 Amazon EC2 Auto Scaling 的[GitHub 儲存庫](#)。

限制

- 在 Auto Scaling 群組的終止政策中，您只能指定一個 Lambda 函數。如果指定了多個終止政策，則必須先指定 Lambda 函數。
- 您可以使用不合格的 ARN (不含字尾) 或具有版本或別名作為字尾的合格 ARN 來引用您的 Lambda 函數。如果使用不合格的 ARN (例如 `function:my-function`)，您必須在函數的未發佈版本上建立資源型政策。如果使用合格的 ARN (例如 `function:my-function:1` 或 `function:my-function:prod`)，您必須在函數的特定發佈版本上建立資源型政策。
- 您不可將合格的 ARN 與 `$LATEST` 字尾搭配使用。如果嘗試新增自訂終止政策，而該政策引用與 `$LATEST` 字尾搭配使用的合格 ARN，則會導致錯誤。
- 輸入資料中提供的執行個體數量限制為 30,000 個執行個體。如果有超過 30,000 個執行個體可以終止，則輸入資料會包括 `"HasMoreInstances": true` 以指出傳回的執行個體數量上限。

- Lambda 函數的執行時間上限為兩秒 (2000 毫秒)。最佳實務是，您應該根據預期的執行時間來設定 Lambda 函數的逾時值。Lambda 函數的預設逾時值為三秒，但此設定可以減少。
- 如果您的執行時間超過 2 秒的限制，則任何擴展動作都會保留，直到執行時間低於此閾值為止。對於執行時間持續較長的 Lambda 函數，請尋找減少執行時間的方法，例如快取可在後續 Lambda 調用期間擷取的結果。
- 不支援跨帳戶 Lambda 函數。用作自訂終止政策的 Lambda 函數必須與 Auto Scaling 群組 AWS 帳戶位於相同的 中。此頁面上的設定指示特別著重於相同帳戶組態。
- 自訂終止政策不適用於運作狀態不佳的執行個體。運作狀態不佳的執行個體一律視為符合終止資格，Lambda 函數只會評估剩餘執行個體的終止順序。

使用執行個體縮減保護來控制執行個體終止

執行個體縮減保護可讓您控制 Amazon EC2 Auto Scaling 可以終止哪些執行個體。此功能的常見使用案例是擴展容器型工作負載。如需詳細資訊，請參閱[設計您的應用程式以正常處理執行個體終止](#)。

根據預設，當您建立 Auto Scaling 群組時，會停用執行個體縮減保護。這表示 Amazon EC2 Auto Scaling 可以終止群組中的任何執行個體。

您可以在 Auto Scaling 群組上啟用執行個體縮減保護設定，一旦執行個體啟動，即可對其進行保護。當執行個體縮減保護啟動時，執行個體的狀態為 InService。然後，若要控制哪些執行個體可以終止，請停用 Auto Scaling 群組中個別執行個體的縮減保護設定。這樣，您可以繼續保護某些執行個體免受不必要的終止。

主題

- [考量事項](#)
- [變更 Auto Scaling 群組的縮減保護](#)
- [變更執行個體的縮減保護](#)

考量事項

以下是使用執行個體縮減保護時的考量：

- 如果 Auto Scaling 群組中的所有執行個體都受到縮減保護，且發生縮減事件，則會減少其所需的容量。不過，在停用執行個體縮減保護設定之前，Auto Scaling 群組無法終止所需的執行個體數量。在 AWS 管理主控台，如果 Auto Scaling 群組中的所有執行個體在發生縮減事件時都受到保護，則 Auto Scaling 群組的活動歷史記錄會包含下列訊息：Could not scale to desired capacity because all remaining instances are protected from scale in.

- 如果您分離受到縮減保護的執行個體，則會遺失其執行個體縮減保護設定。當您再次將執行個體連接到群組時，它會繼承目前執行個體在群組的保護設定中擴展。當 Amazon EC2 Auto Scaling 啟動新的執行個體，或將執行個體從暖集區移至 Auto Scaling 群組時，執行個體會繼承 Auto Scaling 群組的執行個體規模保護設定。
- 執行個體縮減保護並不保護以下的 Auto Scaling 執行個體：
 - 如果執行個體未通過運作狀態檢查，則發生運作狀態檢查取代。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。
 - Spot 執行個體中斷。Spot 執行個體的容量無法再使用或 Spot 價格超過您的最高價時將被終止。
 - 容量區塊保留結束。Amazon EC2 會回收容量區塊執行個體，即使執行個體受到縮減保護。
 - 透過 `terminate-instance-in-auto-scaling-group` 命令手動終止。如需詳細資訊，請參閱[終止 Auto Scaling 群組中的執行個體 \(AWS CLI\)](#)。
 - 透過 Amazon EC2 主控台、CLI 命令和 API 操作手動終止。若要保護 Auto Scaling 執行個體不被手動終止，請啟用 Amazon EC2 終止保護。(這不會阻止 Amazon EC2 Auto Scaling 透過 `terminate-instance-in-auto-scaling-group` 命令終止執行個體或手動終止。) 如需在啟動範本中啟用 Amazon EC2 終止保護的詳細資訊，請參閱[使用進階設定建立啟動範本](#)。

變更 Auto Scaling 群組的縮減保護

您可以啟用或停用 Auto Scaling 群組的執行個體縮減保護設定。當您啟用它時，群組啟動的所有新執行個體都會啟用執行個體縮減保護。

啟用或停用 Auto Scaling 群組的此設定不會影響現有的執行個體。

Console

為新的 Auto Scaling 群組啟用縮減保護

當您建立 Auto Scaling 群組時，在設定群組大小和擴展政策頁面的執行個體縮減保護下，選取啟用執行個體縮減保護核取方塊。

啟用或停用現有群組的縮減保護

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Advanced configurations (進階組態)、Edit (編輯)。

4. 針對執行個體縮減保護，選取或清除啟用執行個體規模保護核取方塊，以視需要啟用或停用此選項。
5. 選擇更新。

AWS CLI

為新的 Auto Scaling 群組啟用縮減保護

使用以下 [create-auto-scaling-group](#) 命令，啟用執行個體縮減保護功能。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --new-  
instances-protected-from-scale-in ...
```

為現有群組啟用縮減保護

使用以下 [update-auto-scaling-group](#) 命令，為指定的 Auto Scaling 群組啟用執行個體縮減保護功能。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --new-  
instances-protected-from-scale-in
```

停用現有群組的縮減保護

使用以下命令來停用所指定群組的執行個體縮減保護。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --no-new-  
instances-protected-from-scale-in
```

變更執行個體的縮減保護

根據預設，執行個體會從本身的 Auto Scaling 群組取得執行個體縮減保護設定。不過，您可以啟用或停用個別執行個體啟動後的執行個體縮減保護。

Console

啟用或停用執行個體的縮減保護

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。

2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Instance management (執行個體管理) 索引標籤的 Instances (執行個體) 中，選取執行個體。
4. 若要啟用執行個體縮減保護，請依序選擇 Actions (動作)、Set scale-in protection (設定縮減保護)。出現提示時，請選擇 Set scale-in protection (設定縮減保護)。
5. 若要停用執行個體縮減保護，請依序選擇 Actions (動作)、Remove scale-in protection (移除縮減保護)。出現提示時，請選擇 Remove scale-in protection (移除縮減保護)。

AWS CLI

啟用執行個體的縮減保護

使用以下 [set-instance-protection](#) 命令，為指定的執行個體啟用執行個體縮減保護。

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-group-name my-asg --protected-from-scale-in
```

停用執行個體的縮減保護

使用以下命令來停用所指定執行個體的縮減保護。

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-group-name my-asg --no-protected-from-scale-in
```

Note

請記住，執行個體縮減保護並不保證在發生人為錯誤時不會終止執行個體，例如，有人使用 Amazon EC2 主控台或手動終止執行個體 AWS CLI。若要保護執行個體不會意外終止，您可以使用 Amazon EC2 終止保護。不過，即使啟用終止保護和執行個體縮減保護，如果運作狀態檢查判斷執行個體狀態不佳或群組本身遭到意外刪除，則儲存到執行個體儲存體的資料仍可能會遺失。與任何環境一樣，最佳實務是經常備份您的資料，或在適合您的業務持續性需求時備份資料。

設計您的應用程式以正常處理執行個體終止

本主題說明您可以用來防止 Amazon EC2 Auto Scaling 群組終止尚未準備好終止的 Amazon EC2 執行個體的功能。根據預設，Auto Scaling 無法查看執行個體上執行的應用程式。它可以在您的應用程式能夠正常關閉或完成其指派的任務之前終止執行個體。這些功能可讓您的應用程式有時間完成進行中的工作、傳輸狀態，或在執行個體終止之前執行清除。您可以根據應用程式的需求，個別或組合使用它們。

這些功能對於具狀態工作負載特別有用，其中機群中的每個執行個體都具有與其他執行個體不同的資料、任務或狀態。在沒有正常關機的情況下終止具狀態的執行個體可能會導致長時間執行的任務從一開始就重新啟動、減少資料備援或資料遺失，以及中斷進行中的交易或運算。若要正常關閉具狀態的執行個體，其工作負載應該耗盡（完成所有目前指派的任務）或傳輸（將任務、資料或組態移至另一個作用中的執行個體）。

目錄

- [終止生命週期關聯](#)
- [執行個體縮減保護](#)
- [自訂終止政策](#)
- [執行個體生命週期政策](#)
- [完全暫停終止](#)
- [限制](#)
- [範例方案](#)

終止生命週期關聯

終止生命週期關聯可延長已選擇終止之 Amazon EC2 執行個體的生命週期。它提供額外的時間來完成目前指派給執行個體的進行中工作，或儲存進度並將工作轉移到另一個執行個體。

對於許多工作負載，終止生命週期關聯可能足以正常關閉為終止選取的執行個體上的應用程式。這是一種盡最大努力的方法，如果捨棄終止生命週期動作，則無法用於防止終止。如果您的工作負載對終止生命週期動作失敗的容錯能力較低，請結合您的終止生命週期關聯來設定[執行個體生命週期政策](#)，以保留執行個體。

若要使用終止生命週期掛鉤，您需要知道何時選取執行個體進行終止。有兩種方式可以得知：

選項	Description	最適合用於	文件的連結
執行個體內部	執行個體中繼資料服務 (IMDS) 是安全的端點，您可以直接從執行個體輪詢執行個體的狀態。如果中繼資料帶回 Terminated，則您的執行個體就會排程終止。	必須在執行個體終止前對執行個體執行動作的應用程式。	擷取目標生命週期狀態
執行個體外部	當執行個體終止時，會產生事件通知。您可以使用 Amazon EventBridge、Amazon SQS、Amazon SNS 或 建立規則 AWS Lambda 來擷取這些事件，並使用 Lambda 函數调用 等回應。	需要在執行個體之外採取動作的應用程式。	設定通知目標

若要使用 lifecycle hook，您還需要知道執行個體何時可以完全終止。Amazon EC2 Auto Scaling 在收到 [CompleteLifecycleAction](#) API 呼叫或逾時經過之前，不會終止執行個體，以先發生者為準。

預設情況下，由於終止 lifecycle hook 的作用，執行個體可以繼續執行一小時 (活動訊號逾時)。如果一小時的時間不足以完成生命週期動作，您可以設定預設逾時。當生命週期動作正在進行時，您可以使用 [RecordLifecycleActionHeartbeat](#) API 呼叫來延長逾時。

如需詳細資訊，請參閱[Amazon EC2 Auto Scaling lifecycle hook](#)。

執行個體縮減保護

您可以使用執行個體縮減保護來控制在縮減事件期間選取要終止的執行個體，尤其是防止正在處理長時間執行任務的執行個體遭到終止。例如，執行容器化工作負載時，通常希望保護所有執行個體，並僅為沒有目前或排程任務的執行個體移除保護。執行個體可以繼續輪詢新任務，並在指派新任務時重新啟用保護。

您可以在 Auto Scaling 群組層級和執行個體層級啟用縮減保護。當您在 Auto Scaling 群組層級啟用縮減保護時，只會在建立新執行個體時受到保護。對於現有的執行個體，您可以個別啟用保護。

應用程式可以從執行個體本身設定保護，或從管理每個執行個體是否可終止的集中式控制平面設定保護。我們建議針對大型機群或需要頻繁切換保護時採用集中式方法，因為它可讓您批次呼叫 [SetInstanceProtection](#)，並避免 API 限流問題。

如需詳細資訊，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。

自訂終止政策

如同執行個體縮減保護，自訂終止政策可協助您防止 Amazon EC2 Auto Scaling 群組終止特定 EC2 執行個體。無論您的自訂終止政策為何，仍可以終止運作狀態不佳的執行個體。

您的 Auto Scaling 群組使用預設終止政策來判斷其先終止的 Amazon EC2 執行個體。如果您想要進一步控制哪些執行個體先終止，您可以使用 Lambda 函數實作自訂終止政策。Auto Scaling 會在需要選取執行個體以進行終止時呼叫此函數，而且只會終止函數傳回的執行個體。如果函數錯誤、逾時或傳回空白清單，Auto Scaling 不會終止任何執行個體，除非執行個體運作狀態不佳。

當您的應用程式可以識別哪些執行個體處於閒置狀態或可安全終止時，自訂終止政策非常有用。這通常需要可跨群組追蹤工作負載的控制平面。

如需詳細資訊，請參閱[使用 Lambda 建立自訂終止政策](#)。

執行個體生命週期政策

執行個體生命週期政策可在捨棄終止生命週期動作時提供 Amazon EC2 Auto Scaling 終止的保護。與單獨使用生命週期掛鉤不同，執行個體生命週期政策旨在確保執行個體在正常關機程序未成功完成時移至保留狀態。

當 Auto Scaling 選取要終止的執行個體時，系統會叫用您設定的終止生命週期掛鉤，而且您的應用程式會開始正常的關閉程序。如果使用 成功完成終止生命週期動作CONTINUE，執行個體會正常終止。不過，如果因任何原因放棄終止生命週期動作，執行個體生命週期政策會將執行個體移至保留狀態，而不是終止它。保留的執行個體不會計入 Auto Scaling 群組所需的容量，因此會自動啟動替換執行個體。除非您使用 [TerminateInstanceInAutoScalingGroup](#) API 手動終止保留的執行個體，否則會針對保留的執行個體及其替換產生標準 Amazon EC2 費用。

若要使用此功能，您必須設定執行個體生命週期政策，並將TerminateHookAbandon保留觸發條件設定為 retain，以及至少一個終止生命週期關聯。由於保留的執行個體會產生持續的 Amazon EC2 成本，且需要手動動作，因此監控至關重要。您應該啟用 CloudWatch 指標，例如 GroupTerminatingRetainedInstances並建立 CloudWatch 警示，以便在執行個體進入保留狀態時提醒您。

如需詳細資訊，請參閱[使用執行個體生命週期政策控制執行個體保留](#)。

完全暫停終止

如果您需要完全控制 Amazon EC2 Auto Scaling 群組中的所有 EC2 執行個體終止，請暫停 Terminate 程序。Amazon EC2 只有在上述選項未提供您服務所需的控制時，才建議使用此選項。透過呼叫 [SuspendProcesses](#) 來暫停 Terminate 程序，您可以防止 Auto Scaling 因故嘗試終止，但使用者對 [TerminateInstanceInAutoScalingGroup API](#) 提出的請求除外。

如需詳細資訊，請參閱[暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。

限制

Important

在 Amazon EC2 Auto Scaling 上設計應用程式以正常處理執行個體終止時，請記住下列限制。

運作狀態不佳的執行個體會略過一些保護

如果執行個體運作狀態不佳，Amazon EC2 Auto Scaling 會開始終止執行個體，即使您有自訂終止政策或縮減保護。防止 Auto Scaling 取代運作狀態不佳執行個體的唯一方法是暫停 HealthCheck、ReplaceUnhealthy 或 Terminate 程序。您可以使用生命週期關聯和執行個體生命週期政策，允許應用程式正常關閉，或複製在終止運作狀態不佳的執行個體之前需要復原的任何資料。

功能	控制運作狀態良好的執行個體	控制運作狀態不佳的執行個體
自訂終止政策	是	否
縮減保護	是	否
暫停 HealthCheck 、 ReplaceUnhealthy 或 Terminate 程序	是	是
lifecycle hook	是	是

功能	控制運作狀態良好的執行個體	控制運作狀態不佳的執行個體
執行個體生命週期政策	是	是

僅生命週期掛鉤並不保證正常關閉

根據預設，終止生命週期掛鉤會盡力運作。如果已捨棄終止生命週期動作，Amazon EC2 Auto Scaling 會繼續立即終止執行個體。您可以結合終止生命週期關聯與執行個體生命週期政策，在捨棄終止生命週期動作時保留執行個體。使用此組合：

- 在 Auto Scaling 觸發執行個體終止且從任何設定的 Elastic Load Balancing 負載平衡器耗盡完成後，您的終止生命週期掛鉤會嘗試正常關閉應用程式。
- 如果由於任何原因放棄終止生命週期動作，執行個體會移至保留狀態，而不是終止。
- 保留的執行個體會保持在目前的 Amazon EC2 狀態，可讓您手動完成關機程序或調查失敗。
- 您可以在完成必要的動作後呼叫 [TerminateInstanceInAutoScalingGroup](#) API，手動終止保留的執行個體。

如需詳細資訊，請參閱[使用執行個體生命週期政策控制執行個體保留](#)。

某些執行個體市場選項可能會在有限通知的情況下中斷

如果您在 Auto Scaling 群組中使用 Spot 執行個體和中斷容量保留等執行個體市場選項，Amazon EC2 可以隨時中斷並回收您的執行個體。這些中斷會略過所有 Amazon EC2 Auto Scaling 保護機制，包括：

- 終止生命週期關聯
- 執行個體縮減保護
- 自訂終止政策
- 執行個體生命週期政策
- 暫停的程序

當 Spot 執行個體收到中斷通知時，您有大約兩分鐘的時間來執行正常的關機任務。雖然您可以使用終止生命週期關聯來回應 Spot 執行個體中斷，但即使生命週期關聯仍在進行中，執行個體仍會在兩分鐘時段結束時強制終止。執行個體生命週期政策也無法防止 Spot 執行個體中斷。

如需處理 Spot 執行個體中斷的詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [Spot 執行個體中斷](#)和 Amazon EC2 Spot 的最佳實務。 [Amazon EC2](#) Amazon EC2

直接 Amazon EC2 終止會略過所有保護

如果您使用 Amazon EC2 [TerminateInstances API](#) 直接終止 Auto Scaling 群組中的執行個體，則終止會略過所有 Amazon EC2 Auto Scaling 保護機制。

若要終止 Auto Scaling 群組中的執行個體，同時遵守您設定的保護，請改用 [TerminateInstanceInAutoScalingGroup](#) API。

範例方案

當您使用 Amazon EC2 Auto Scaling 時，可以選擇 Auto Scaling 代表您處理多少機群管理，以及您對 EC2 執行個體終止決策保留多少直接控制。工作負載對執行個體終止的敏感度越高，您可能想要保留的控制權就越高。下列範例說明不同容錯程度和建議組態的工作負載：

範例 1：分散式資料庫節點（低容錯能力）

您會執行分散式資料庫，其中每個 EC2 執行個體會保留複寫係數為 3 的資料分割區。遺失保留相同分割區複本的多個執行個體可能會導致資料遺失或使該分割區無法使用。

挑戰：Auto Scaling 終止執行個體的速度可能比將資料重新複寫到其他節點的速度更快，而且終止可能會降低您的容量，低於維護複寫因素所需的容量。

請考慮下列組態：

- 在所有資料庫執行個體上啟用[執行個體縮減保護](#)；只有在確認資料已在別處安全複寫之後，才能以程式設計方式將其移除。
- 設定具有延長逾時的[終止生命週期關聯](#)，並結合執行個體生命週期政策，以允許資料傳輸完成。
- 設定運作狀態最低百分比為 100% 的[執行個體維護政策](#)，以維持所需的容量。

範例 2：長時間執行的任務處理（中等容錯能力）

您有 Amazon SQS 佇列，可收集長時間執行任務的傳入訊息。當新訊息送達時，EC2 執行個體會擷取訊息並啟動需要 3 小時才能處理的任務。隨著佇列的成長，Auto Scaling 會根據您的擴展政策新增執行個體。隨著佇列縮減，Auto Scaling 會終止執行個體。

挑戰：Auto Scaling 可能會終止處理任務需要 3 小時的執行個體，而不是閒置的執行個體。任務可以在另一個執行個體上重新啟動，但您會失去重大進度。

請考慮下列組態：

- 設定[自訂終止政策](#)，先排定終止閒置執行個體的優先順序。
- 使用[終止生命週期關聯](#)來允許進行中的任務完成。
- 當[執行個體啟動任務時，以程式設計方式啟用執行個體縮減保護](#)，並在任務完成時將其移除。

範例 3：適用於測試環境的工作者機群（高容錯能力）

您可以執行執行自動化測試、CI/CD 管道任務或開發工作負載的 EC2 執行個體機群。這些工作者執行個體會從佇列中提取任務，如果任務失敗，則可以重新產生測試結果。

挑戰：測試任務可能會在縮減事件期間中斷，但由於測試可以在不影響的情況下重試，因此您想要最佳化成本和簡單性，而不是零中斷可用性。

請考慮下列組態：

- 使用[終止生命週期關聯](#)來允許處理中的請求完成。
- 考慮使用具有容量最佳化配置策略的[Spot 執行個體](#)，以進一步降低成本。

暫停和繼續 Amazon EC2 Auto Scaling 程序

本主題說明如何暫停，然後繼續 Auto Scaling 群組的一或多個程序，以暫時停用特定操作。

當您需要在不受擴展政策或排程動作干擾的情況下調查或疑難排解問題時，暫停程序會很有用。這也有助於防止 Amazon EC2 Auto Scaling 在您變更 Auto Scaling 群組時，將執行個體標示為運作狀態不佳並加以取代。

主題

- [程序類型](#)
- [暫停程序的考量事項](#)
- [暫停程序](#)
- [繼續程序](#)
- [暫停程序如何影響其他程序](#)

Note

除了您啟動的暫停作業之外，Amazon EC2 Auto Scaling 也可以針對一直無法啟動執行個體的 Auto Scaling 群組，暫停其處理程序。這也稱為管理暫停。管理暫停最常應用在嘗試啟動

執行個體超過 24 小時，但是皆失敗的 Auto Scaling 群組。您可以繼續執行因管理原因而遭到 Amazon EC2 Auto Scaling 暫停的處理程序。

程序類型

暫停繼續功能支援以下程序：

- **Launch** – 當群組向外擴展，或當 Amazon EC2 Auto Scaling 因其他原因選擇啟動執行個體時，例如將執行個體新增至暖集區時，將執行個體新增至 Auto Scaling 群組。
- **Terminate** – 當群組縮減時，或當 Amazon EC2 Auto Scaling 因其他原因選擇終止執行個體時，例如當執行個體因超過其最長生命週期或運作狀態檢查失敗而終止時，從 Auto Scaling 群組移除執行個體。
- **AddToLoadBalancer** – 啟動執行個體時，將執行個體新增至連接的負載平衡器目標群組或 Classic Load Balancer。如需詳細資訊，請參閱[使用 Elastic Load Balancing 在 Auto Scaling 群組中分配傳入應用程式流量](#)。
- **AlarmNotification** – 接受來自與動態擴展政策相關聯的 CloudWatch 警示的通知。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 動態擴展](#)。
- **AZRebalance** – 當群組變得不平衡時，平衡群組中所有指定可用區域的 EC2 執行個體數量，例如，當先前無法使用的可用區域返回運作狀態良好時。如需詳細資訊，請參閱[重新平衡活動](#)。
- **HealthCheck** – 檢查執行個體的運作狀態，如果 Amazon EC2 或 Elastic Load Balancing 通知 Amazon EC2 Auto Scaling 執行個體運作狀態不佳，則將執行個體標記為運作狀態不佳。此處理程序會覆寫您手動設定執行個體的運作狀態。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。
- **InstanceRefresh** – 使用執行個體重新整理功能終止和取代執行個體。如需詳細資訊，請參閱[使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。
- **ReplaceUnhealthy** – 終止標示為運作狀態不佳的執行個體，然後建立新的執行個體來取代它們。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。
- **ScheduledActions** – 執行您建立的排程擴展動作，或在您建立 AWS Auto Scaling 擴展計劃並開啟預測擴展時為您建立的排程擴展動作。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 排程擴展](#)。

暫停程序的考量事項

暫停處理程序之前，請考慮下列事項：

- 暫停 AlarmNotification 可讓您暫時停止群組的目標追蹤、步驟及簡單擴展政策，而不必刪除擴展政策或其關聯的 CloudWatch 警示。若要暫時停止個別擴展政策，請參閱 [停用 Auto Scaling 群組的擴展政策](#)。
- 您可以選擇暫停 HealthCheck 和 ReplaceUnhealthy 程序以重新啟動執行個體，而不讓 Amazon EC2 Auto Scaling 根據執行個體的運作狀態檢查終止執行個體。不過，如果您需要 Amazon EC2 Auto Scaling 繼續對其餘執行個體執行運作狀態檢查，請改用待命功能。如需詳細資訊，請參閱 [從 Auto Scaling 群組暫時移除執行個體](#)。
- 如果暫停 Launch 和 Terminate 處理程序或 AZRebalance，然後變更 Auto Scaling 群組 (例如，透過分開執行個體或變更指定的可用區域)，您的群組會在可用區域間呈現不平衡狀態。若該狀況發生，在您繼續執行暫停的處理程序後，Amazon EC2 Auto Scaling 會在可用區域之間逐步重新分配執行個體。
- 如果您暫停 Terminate 程序，仍然可以使用 [delete-auto-scaling-group](#) 命令搭配強制刪除選項來強制終止執行個體。
- 暫停 Terminate 程序僅適用於目前處於 InService 狀態的執行個體。它不會阻止在其他狀態下終止執行個體，例如 Pending，或無法從待命中正常恢復的執行個體。
- 當呼叫中存在使用 AWS CLI 或 SDKs 描述 Auto Scaling 群組時，可以忽略 RemoveFromLoadBalancerLowPriority 此程序。此程序已棄用，且僅為了回溯相容性而予以保留。

暫停程序

若要暫停 Auto Scaling 群組的程序，請使用下列其中一種方法：

Console

暫停處理程序

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Advanced configurations (進階組態)、Edit (編輯)。
4. 對於 Suspended processes (暫停處理程序)，選擇要暫停的處理程序。
5. 選擇更新。

AWS CLI

使用以下 [suspend-processes](#) 命令來暫停個別處理程序。

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg --scaling-processes HealthCheck ReplaceUnhealthy
```

若要暫停所有處理程序，請忽略 `--scaling-processes` 選項，如下所示。

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg
```

繼續程序

若要恢復 Auto Scaling 群組的暫停程序，請使用下列其中一種方法：

Console

恢復暫停的處理程序

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Advanced configurations (進階組態)、Edit (編輯)。
4. 對於 Suspended processes (暫停處理程序)，請移除暫停的處理程序。
5. 選擇更新。

AWS CLI

若要繼續暫停的程序，請使用下列 [resume-processes](#) 命令。

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg --scaling-processes HealthCheck
```

若要繼續執行所有暫停的處理程序，請忽略 `--scaling-processes` 選項，如下所示。

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg
```

暫停程序如何影響其他程序

下列各節說明個別暫停不同程序時會發生的情況。

主題

- [Launch 已暫停](#)
- [Terminate 已暫停](#)
- [AddToLoadBalancer 已暫停](#)
- [AlarmNotification 已暫停](#)
- [AZRebalance 已暫停](#)
- [HealthCheck 已暫停](#)
- [InstanceRefresh 已暫停](#)
- [ReplaceUnhealthy 已暫停](#)
- [ScheduledActions 已暫停](#)
- [其他考量](#)

Launch 已暫停

- AlarmNotification 仍在作用中，但 Auto Scaling 群組無法針對違規警示啟動橫向擴展活動。
- ScheduledActions 在作用中，但 Auto Scaling 群組無法針對任何發生的排定動作啟動擴展活動。
- AZRebalance 停止重新平衡群組。
- ReplaceUnhealthy 繼續終止運作狀態不佳的執行個體，但不啟動替換。當您繼續 Launch 處理程序時，Amazon EC2 Auto Scaling 會立即替換在 Launch 暫停期間終止的任何執行個體。
- InstanceRefresh 不會取代執行個體。

Terminate 已暫停

- AlarmNotification 仍然作用中，但 Auto Scaling 群組無法針對違規的警示啟動縮減活動。
- ScheduledActions 處於作用中狀態，但 Auto Scaling 群組無法針對發生的任何排程動作啟動縮減活動。
- AZRebalance 仍在作用中，但無法正常運作。它可以啟動新的執行個體，而不會終止舊的執行個體。您的 Auto Scaling 群組最多可以將容量增加到超過上限 10%，因為在進行重新平衡活動時，可

允許此種情況發生。您的 Auto Scaling 群組可以維持超過其容量上限，直到您繼續執行 Terminate 程序為止。

- **ReplaceUnhealthy** 非作用中，但不 **HealthCheck**。當 Terminate 繼續時，ReplaceUnhealthy 處理程序立即開始執行。如果任何執行個體在 Terminate 暫停時標示為運作狀態不佳，則會立即替換執行個體。
- **InstanceRefresh** 不會取代執行個體。

AddToLoadBalancer 已暫停

- Amazon EC2 Auto Scaling 會啟動執行個體，但不將其新增至負載平衡器目標群組或 Classic Load Balancer。如果您繼續 AddToLoadBalancer 程序，當執行個體啟動時，會將新增的執行個體連接至負載平衡器。不過，當程序暫停時，不會新增已啟動的執行個體。您必須手動註冊這些執行個體。

AlarmNotification 已暫停

- 違反 CloudWatch 警示閾值時，Amazon EC2 Auto Scaling 不會叫用擴展政策。在您繼續 AlarmNotification 時，Amazon EC2 Auto Scaling 會以目前違反的警示閾值考量政策。

AZRebalance 已暫停

- Amazon EC2 Auto Scaling 並未嘗試在某些事件後重新分配執行個體。不過，如果發生橫向擴展或縮減事件，擴展程序仍會嘗試平衡可用區域。例如，在向外擴展期間，它會在可用區域中以最少的執行個體啟動執行個體。如果群組在 AZRebalance 暫停時變成不平衡，且您繼續此群組，Amazon EC2 Auto Scaling 則會嘗試重新平衡群組。它會先呼叫 Launch，然後再呼叫 Terminate。
- AZRebalance 暫停時，暖集區不會受到影響。

HealthCheck 已暫停

- Amazon EC2 Auto Scaling 會因 EC2 和 Elastic Load Balancing 運作狀態而停止將執行個體標示為運作狀態不佳。您的自訂運作狀態檢查會繼續正常運作。暫停 HealthCheck 後，如果需要，您可以手動設定群組中執行個體的運作狀態，並讓 ReplaceUnhealthy 替換。

InstanceRefresh 已暫停

- Amazon EC2 Auto Scaling 會因執行個體重新整理而停止取代執行個體。如果正在執行執行個體重新整理，則會暫停操作 (不會取消操作)。

ReplaceUnhealthy 已暫停

- Amazon EC2 Auto Scaling 會停止替換標示為運作狀態不佳的執行個體。未通過 EC2 或 Elastic Load Balancing 運作狀態檢查的執行個體仍會被標示為運作狀態不佳。一旦繼續 ReplaceUnhealthy 處理程序，Amazon EC2 Auto Scaling 便會立即替換在此處理程序遭到暫停時標示為運作狀態不佳的執行個體。ReplaceUnhealthy 程序會先呼叫 Terminate，然後再呼叫 Launch。

ScheduledActions 已暫停

- Amazon EC2 Auto Scaling 不會執行預定在暫停期間執行的已排程動作。當您繼續 ScheduledActions 時，Amazon EC2 Auto Scaling 僅會考量尚未超過其排程時間的排程動作。

其他考量

此外，當 Launch 或 Terminate 暫停時，下列功能可能無法正常運作：

- 執行個體生命週期上限 – Launch 或 Terminate 暫停時，執行個體生命週期上限功能無法取代任何執行個體。
- Spot 執行個體中斷 – 如果 Terminate 已暫停，且 Auto Scaling 群組具有 Spot 執行個體，則仍然可以在 Spot 容量不再可用時終止。Launch 遭到暫停時，Amazon EC2 Auto Scaling 無法在重新可用時從其他 Spot 執行個體集區，或從相同的 Spot 執行個體集區啟動替換執行個體。
- 容量重新平衡 – 如果 Terminate 已暫停，而您使用容量重新平衡來處理 Spot 執行個體中斷，Amazon EC2 Spot 服務仍然可以在 Spot 容量不再可用時終止執行個體。如果暫停 Launch，則 Amazon EC2 Auto Scaling 無法在重新可用時從其他 Spot 執行個體集區，或從相同的 Spot 執行個體集區啟動替換執行個體。
- 連接和分離執行個體 – 暫停 Launch 和 Terminate 時，您可以分離連接至 Auto Scaling 群組的執行個體，但在 Launch 暫停時，您無法將新執行個體連接至群組。
- 待命執行個體 – 暫停 Launch 和 Terminate 時，您可以將執行個體置於 Standby 狀態，但在 Launch 暫停時，您無法將處於 Standby 狀態的執行個體傳回 服務。

監控 Amazon EC2 Auto Scaling 群組

監控是維護 Amazon EC2 Auto Scaling 和 AWS 雲端 解決方案可靠性、可用性和效能的重要部分。AWS 提供下列監控工具來監看 Amazon EC2 Auto Scaling、在發生錯誤時回報，並適時採取自動動作：

運作狀態檢查

Amazon EC2 Auto Scaling 會定期對 Auto Scaling 群組中的執行個體進行運作狀態檢查。若執行個體未通過運作狀態檢查，則會標示為運作狀態不佳，並在 Amazon EC2 Auto Scaling 啟動新執行個體以取代時終止該執行個體。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

AWS Health 儀板表

Health 儀板表 會顯示資訊，並提供由 AWS 資源運作狀態變更所叫用的通知。該資訊以兩種方式呈現：儀表板 (依類別顯示最近和近期事件) 和完整的事件日誌 (顯示過去 90 天內的所有事件)。如需詳細資訊，請參閱[Health 儀板表 Amazon EC2 Auto Scaling 的通知](#)。

CloudTrail

透過 AWS CloudTrail，您可以追蹤由 或代表您對 Amazon EC2 Auto Scaling API 發出的呼叫 AWS 帳戶。CloudTrail 會將資訊存放到您所指定的 Amazon S3 儲存貯體的日誌檔案中。您可以使用這些日誌檔案來監控 Auto Scaling 群組的活動。日誌包含提出了什麼請求、提出請求的來源 IP 地址、提出請求的人員、提出請求的時間等。如需詳細資訊，請參閱[使用 記錄 Amazon EC2 Auto Scaling API 呼叫 AWS CloudTrail](#)。

Amazon EC2 執行個體的日誌集合

您可以使用 CloudWatch 從作業系統收集 EC2 執行個體的日誌。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[使用 CloudWatch 代理程式從 Amazon EC2 執行個體和內部部署伺服器收集指標和日誌](#)以及[檢視傳送到 CloudWatch 日誌的日誌資料](#)。如需其他可協助您記錄和收集工作負載相關資料之 AWS 服務的相關資訊，請參閱《方案指南》中的[應用程式擁有者的記錄和監控](#)指南。AWS

Amazon CloudWatch

Amazon CloudWatch 可協助您分析日誌，並即時監控 AWS 資源與託管應用程式的指標。您可以收集和追蹤指標、建立自訂儀板表，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動

作。例如，當網路活動突然高於或低於指標的預期值時，您就會收到通知。如需有關使用此服務監控 Auto Scaling 群組和執行個體的指標的詳細資訊，請參閱 [監控 Auto Scaling 群組和執行個體的 CloudWatch 指標](#)。

CloudWatch 也會追蹤 Amazon EC2 Auto Scaling 的 AWS API 用量指標。您可以使用這些指標來設定警示，即可在 API 呼叫量違反您定義的閾值時提醒您。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [AWS 用量指標](#)。

AWS Compute Optimizer

Compute Optimizer 提供 Amazon EC2 執行個體推薦服務，可協助您決定是否要移至新的執行個體類型。該工具會分析 Auto Scaling 群組的執行個體是否為最佳類型，並會產生建議協助您節省成本並改善工作負載的效能。如需詳細資訊，請參閱 [使用 取得執行個體類型建議 AWS Compute Optimizer](#)。

Amazon EventBridge

Amazon EventBridge 為無伺服器事件匯流排服務，可讓您輕鬆將應用程式與來自各種來源的資料互相連線。EventBridge 會從您自己的應用程式、Software-as-a-Service(SaaS) 應用程式 AWS 和服務提供即時資料串流，並將該資料路由到 Lambda 等目標。這可以讓您監控在服務中發生的事件，並建置事件導向的架構。如需詳細資訊，請參閱 [使用 EventBridge 處理 Auto Scaling 事件](#)。

AWS Security Hub CSPM

使用 [AWS Security Hub CSPM](#) 監控您 Amazon EC2 Auto Scaling 的使用狀況，因為這關係到安全最佳實務。Security Hub CSPM 使用偵測性安全控制來評估資源組態和安全標準，以協助您符合各種合規架構。如需使用 Security Hub CSPM 評估 Amazon EC2 Auto Scaling 資源的詳細資訊，請參閱 AWS Security Hub 《使用者指南》中的 [Amazon EC2 Auto Scaling 控制項](#)。

Amazon Simple Notification Service

您可以設定 Auto Scaling 群組，以便在 Amazon EC2 Auto Scaling 啟動或終止執行個體時傳送 Amazon SNS 通知。如需詳細資訊，請參閱 [Amazon EC2 Auto Scaling 的 Amazon SNS 通知選項 Amazon EC2 Auto Scaling](#)。

Auto Scaling 群組中的執行個體運作狀態檢查

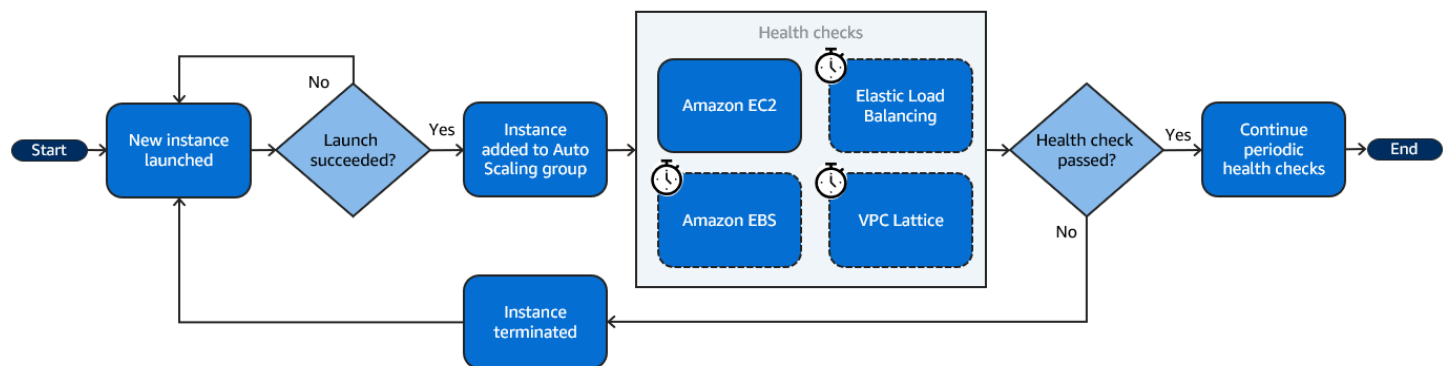
Amazon EC2 Auto Scaling 會持續監控 Auto Scaling 群組中執行個體的運作狀態，以維持所需的容量。

Auto Scaling 群組中的所有執行個體都以 Healthy 狀態開頭。除非 Amazon EC2 Auto Scaling 收到通知說執行個體運作狀態不佳，否則假定執行個體都處於良好狀態。當執行個體運作狀態不佳且必須取代時，它可以接收來自各種來源的通知。這些來源包括：

- Amazon EC2
- Elastic Load Balancing
- VPC Lattice
- Amazon EBS
- 您定義的自訂運作狀態檢查

當 Amazon EC2 Auto Scaling 判斷 InService 執行個體運作狀態不佳時，它會將其取代為新的執行個體，以維持群組所需的容量。新執行個體將使用 Auto Scaling 群組的目前設定及其關聯的啟動範本或啟動組態進行啟動。

下列流程圖說明在 Auto Scaling 群組中啟動新執行個體的程序。首先啟動執行個體。如果啟動成功，執行個體會新增至 Auto Scaling 群組。然後，Amazon EC2 Auto Scaling 會使用內建的 Amazon EC2 狀態檢查對執行個體執行運作狀態檢查，並在寬限期後執行您為群組啟用的任何選用運作狀態檢查。這些運作狀態檢查會定期繼續進行。如果任何運作狀態檢查失敗，則會取代執行個體。



當執行個體意外終止時，也可能發生運作狀態不佳的執行個體，例如因 Spot 執行個體中斷或使用者手動終止。同樣地，在這些情況下，Amazon EC2 Auto Scaling 會自動啟動替換執行個體，以維持所需的容量。

目錄

- [關於 Auto Scaling 群組的運作狀態檢查](#)
- [設定 Auto Scaling 群組的運作狀態檢查寬限期。](#)
- [使用運作狀態檢查監控 Amazon EBS 磁碟區受損的 Auto Scaling 執行個體](#)
- [為您的 Auto Scaling 群組設定自訂運作狀態檢查](#)

- [檢視運作狀態檢查失敗的原因](#)
- [對 Amazon EC2 Auto Scaling 中運作狀態不佳的執行個體進行故障診斷](#)

關於 Auto Scaling 群組的運作狀態檢查

本主題提供可用運作狀態檢查類型的概觀，並說明整合 Amazon EC2 Auto Scaling 運作狀態檢查與應用程式的重要考量。

目錄

- [運作狀態檢查類型](#)
- [Amazon EC2 運作狀態檢查](#)
- [Elastic Load Balancing 運作狀態檢查](#)
- [VPC Lattice 運作狀態檢查](#)
- [Amazon EC2 Auto Scaling 如何將停機時間降至最低](#)
- [暖集區中執行個體的運作狀態檢查](#)
- [運作狀態檢查注意事項](#)

運作狀態檢查類型

Amazon EC2 Auto Scaling 可以使用下列一或多個運作狀態檢查來判斷 InService 執行個體的運作狀態：

運作狀態檢查類型	它檢查的服務項目
Amazon EC2 狀態檢查和排程事件	• 檢查執行個體是否正在執行。 • 檢查是否有可能損害執行個體的基礎硬體或軟體問題。 這是 Auto Scaling 群組的預設運作狀態檢查類型。
Elastic Load Balancing 運作狀態檢查	• 檢查負載平衡器是否將執行個體報告為正常運作，確認執行個體是否可以處理請求。 若要執行此運作狀態檢查類型，您必須為 Auto Scaling 群組將其開啟。

運作狀態檢查類型	它檢查的服務項目
VPC Lattice 運作狀態檢查	檢查 VPC Lattice 是否將執行個體報告為正常運作，確認執行個體是否可以處理請求。 若要執行此運作狀態檢查類型，您必須為 Auto Scaling 群組將其開啟。
Amazon EBS 運作狀態檢查	檢查 EBS 磁碟區是否可連線並傳遞 I/O 狀態檢查。 若要執行此運作狀態檢查類型，您必須為 Auto Scaling 群組將其開啟。
自訂運作狀態檢查	根據您的自訂運作狀態檢查，檢查可能表示執行個體運作狀態問題的任何其他問題。

Amazon EC2 運作狀態檢查

在執行個體啟動後，該執行個體會連接至 Auto Scaling 群組，並進入 InService 狀態。如需 Auto Scaling 群組中執行個體不同生命週期的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 執行個體生命週期](#)。

Amazon EC2 Auto Scaling 會定期檢查 Auto Scaling 群組內所有執行個體的運作狀態，藉此確認執行個體是否正在執行且狀態良好。

狀態檢查

Amazon EC2 Auto Scaling 使用 Amazon EC2 執行個體狀態檢查和系統狀態檢查的結果來判斷執行個體的運作狀態。如果執行個體的状态是 running 以外的任何 Amazon EC2 狀態，或是狀態檢查的狀態變為 impaired，則 Amazon EC2 Auto Scaling 會認定此執行個體運作狀態不佳，並將其取代。這包括執行個體具有下列任何狀態時：

- stopping
- stopped
- shutting-down
- terminated

Amazon EC2 狀態檢查不需要任何特殊組態，並且會永遠啟用。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[狀態檢查類型](#)。

Important

Amazon EC2 Auto Scaling 允許狀態檢查偶爾失敗，而不採取任何動作。當狀態檢查失敗時，Amazon EC2 Auto Scaling 會等待幾分鐘 AWS 讓修正問題。當執行個體狀態檢查的狀態變為 `impaired` 時，其不會立即將執行個體標示為 `Unhealthy`。此外，EC2 Auto Scaling 不會將執行個體標記為 `Unhealthy` 狀態檢查傳回 `insufficient-data`。

然而，如果 Amazon EC2 Auto Scaling 偵測到執行個體不再處於 `running` 狀態，但是，此情況可視為立即失敗。在這種情況下，它會立即將執行個體標記為 `Unhealthy` 並取代它。

排程事件

Amazon EC2 偶爾可以將您執行個體上的事件安排在特定時間戳之後執行。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[執行個體的已排程事件](#)。

若您其中一個執行個體受到排程事件影響，Amazon EC2 Auto Scaling 會認定此執行個體運作狀態不佳，並會進行取代。在達到時間戳記中指定的日期和時間之前，執行個體不會開始關閉。

Elastic Load Balancing 運作狀態檢查

當您開啟 Auto Scaling 群組的 Elastic Load Balancing 運作狀態檢查時，Amazon EC2 Auto Scaling 可以使用這些運作狀態檢查的結果來判斷執行個體的運作狀態。

在您可以為 Auto Scaling 群組開啟 Elastic Load Balancing 運作狀態檢查之前，您必須設定 Elastic Load Balancing 負載平衡器，並為其設定運作狀態檢查，以判斷您的執行個體是否正常運作。如需詳細資訊，請參閱[準備連接 Elastic Load Balancing 負載平衡器](#)。

將負載平衡器連接到 Auto Scaling 群組之後，會發生下列情況：

- Amazon EC2 Auto Scaling 會向負載平衡器註冊 Auto Scaling 群組中的執行個體。
- 在執行個體完成註冊後，其會進入 `InService` 狀態，並可與負載平衡器搭配使用。

預設情況下，Amazon EC2 Auto Scaling 會忽略 Elastic Load Balancing 運作狀態檢查的結果。開啟 Auto Scaling 群組的這些運作狀態檢查後，當 Elastic Load Balancing 將已註冊的執行個體報告為 `Unhealthy`，Amazon EC2 Auto Scaling 會在下次定期運作狀態檢查 `Unhealthy` 時標記執行個體並加以取代。

如果為負載平衡器啟用連接耗盡 (取消註冊的延遲)，Amazon EC2 Auto Scaling 會等待傳送中的請求完成或最長逾時過期 (兩者擇一)，再終止運作狀態不佳的執行個體。

Note

如需如何連接負載平衡器並開啟 Auto Scaling 群組 Elastic Load Balancing 運作狀態檢查的說明，請參閱 [將 Elastic Load Balancing 負載平衡器連接至 Auto Scaling 群組](#)。

當您開啟群組的 Elastic Load Balancing 運作狀態檢查時，Amazon EC2 Auto Scaling 可以取代 Elastic Load Balancing 回報為運作狀態不佳的執行個體，但僅限於負載平衡器處於 InService 狀態之後。如需詳細資訊，請參閱[確認負載平衡器的連接狀態](#)。

VPC Lattice 運作狀態檢查

預設情況下，Amazon EC2 Auto Scaling 會忽略 VPC Lattice 運作狀態檢查的結果。您可以選擇性地為 Auto Scaling 群組開啟這些運作狀態檢查。之後，當 VPC Lattice 將註冊的執行個體報告為 Unhealthy，Amazon EC2 Auto Scaling 會在下次定期執行運作狀態檢查時將執行個體標示為 Unhealthy 並將其取代。註冊執行個體然後檢查其健康狀態的程序，與 Elastic Load Balancing 運作狀態檢查的程序相同。

Note

如需如何連接 VPC Lattice 目標群組和開啟 Auto Scaling 群組 VPC Lattice 運作狀態檢查的說明，請參閱 [將 VPC Lattice 目標群組連接到 Auto Scaling 群組](#)。

當您開啟群組的 VPC Lattice 運作狀態檢查時，Amazon EC2 Auto Scaling 可以取代 VPC Lattice 回報為運作狀態不佳的執行個體，但僅限於目標群組處於 InService 狀態之後。如需詳細資訊，請參閱[確認 VPC Lattice 目標群組的連接狀態](#)。

Amazon EC2 Auto Scaling 如何將停機時間降至最低

根據預設，新的執行個體會在您現有的執行個體終止時佈建，這可能會阻止接受新的請求，直到新的執行個體完全運作為止。

如果 Amazon EC2 Auto Scaling 判斷任何執行個體不再執行（或使用 Unhealthy [set-instance-health](#) 命令標記），則會立即取代它們。但是，如果發現其他執行個體運作狀態不佳，Amazon EC2 Auto Scaling 會使用下列方法來從失敗中復原。此方法最大限度地減少因臨時問題或設定錯誤的運作狀態檢查而可能導致的停機時間。

- 如果擴展活動正在進行中，且 Auto Scaling 群組低於其所需的容量達 10% 或更高，Amazon EC2 Auto Scaling 會等待進行中的擴展活動，然後再取代運作狀態不佳的執行個體。
- 在橫向擴展時，Amazon EC2 Auto Scaling 會等待執行個體通過初始運作狀態檢查。它還會等待預設執行個體完成暖機，以確保新的執行個體準備就緒。
- 執行個體完成暖機後，且群組已上升至所需容量的 90% 以上，Amazon EC2 Auto Scaling 會取代運作狀態不佳的執行個體，如下所示：
 - Amazon EC2 Auto Scaling 最高一次僅取代該群組所需容量的 10%。它會執行此操作，直到所有運作狀態不佳的執行個體完成替換。
 - 取代執行個體時，它會等待新執行個體傳遞初始運作狀態檢查。它還會等待預設執行個體完成暖機，然後再繼續作業。

Note

- 如果 Auto Scaling 群組的大小小到 10% 的結果值小於 1，Amazon EC2 Auto Scaling 會取代運作狀態不佳的執行個體。這可能會導致群組產生一些停機時間。
- 您可以透過[設定執行個體維護政策](#)來變更 Auto Scaling 取代運作狀態不佳執行個體的速率，以修改預設的 10% 值。不過，Auto Scaling 可能仍會調節將執行個體標記為運作狀態不佳的速率。

例如，如果 Elastic Load Balancing 運作狀態檢查報告 Auto Scaling 群組中的所有執行個體運作狀態不佳，且負載平衡器處於 InService 狀態，則 Amazon EC2 Auto Scaling 可能會一次將運作狀態不佳的執行個體標記為較少。這可能導致一次替換的執行個體比在其他案例中套用的 10% 少得多。這可讓您有時間修正問題，而無需 Amazon EC2 Auto Scaling 自動終止整個群組。

暖集區中執行個體的運作狀態檢查

Amazon EC2 Auto Scaling 也會對暖集區中的執行個體執行運作狀態檢查。如需詳細資訊，請參閱[檢視運作狀態檢查狀態和運作狀態檢查失敗的原因](#)。

運作狀態檢查注意事項

以下是使用 Amazon EC2 Auto Scaling 運作狀態檢查時的考量事項。

- 如果您需要對正在終止的執行個體或正在啟動的執行個體做些什麼，可以使用 lifecycle hook。使用這些掛鉤，您可以執行自訂動作，如 Amazon EC2 Auto Scaling 啟動或終止執行個體。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling lifecycle hook](#)。
- Amazon EC2 Auto Scaling 不提供從運作狀態檢查中移除 Amazon EC2 狀態檢查和排程事件的方法。如果不要取代執行個體，建議您暫停任何個別 Auto Scaling 群組的 ReplaceUnhealthy 和 HealthCheck 程序。如需詳細資訊，請參閱[暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。
- 若要手動將運作不佳的執行個體的運作狀態設回 Healthy，您可以試著使用 [set-instance-health](#) 命令。如果您看見錯誤，這可能是因為執行個體已終止。通常，使用 [set-instance-health](#) 命令將執行個體的運作狀態設回 Healthy 的作法，只適用於 ReplaceUnhealthy 程序或 Terminate 程序已暫停處理的情況。
- 如果您需要對執行個體進行故障診斷，而不會受到運作狀態檢查的干擾，您可以將執行個體置於 Standby 狀態。在您讓執行個體恢復服務之前，Amazon EC2 Auto Scaling 不會對處於 Standby 狀態的執行個體執行運作狀態檢查。如需詳細資訊，請參閱[從 Auto Scaling 群組暫時移除執行個體](#)。
- 當您的執行個體終止後，所有與彈性 IP 地址相關聯的設定，都會解除，且不會自動關聯到新的執行個體。您必須手動將這些彈性 IP 地址關聯到新的執行個體，或使用 lifecycle hook 型解決方案自動進行。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[彈性 IP 位址](#)。
- 同樣地，當您的執行個體終止後，其連接的 EBS 磁碟區會分開 (或刪除，具體取決於磁碟區的 DeleteOnTermination 屬性)。您必須手動將這些 EBS 磁碟區連接到新的執行個體，或使用 lifecycle hook 型解決方案自動進行。如需詳細資訊，請參閱「Amazon EBS 使用者指南」中的[將 Amazon EBS 磁碟區連接到執行個體](#)。

設定 Auto Scaling 群組的運作狀態檢查寬限期。

當 Amazon EC2 Auto Scaling 運作狀態檢查確定 InService 執行個體運作狀態不佳時，便會以新的執行個體取代之。運作狀態檢查寬限期指定在發現新執行個體運作狀態不佳時終止服務之前保持服務的最短時間 (以秒為單位)。

使用案例範例可能是：需要 Amazon EC2 Auto Scaling 避免在 Elastic Load Balancing 運作狀態檢查失敗且原因為執行個體仍在初始化時採取動作。Elastic Load Balancing 運作狀態檢查會平行執行，並從執行個體向負載平衡器註冊時開始。寬限期可防止 Amazon EC2 Auto Scaling 標記新啟動的執行個體 Unhealthy，並在它們進入 InService 狀態後未立即通過這些運作狀態檢查時，不必要地終止它們。

在主控台中，根據預設，當您建立 Auto Scaling 群組時，運作狀態檢查的寬限期為 300 秒。當您使用 AWS CLI 或 SDK 建立 Auto Scaling 群組時，其預設值為 0 秒。0 值表示關閉運作狀態檢查寬限期。

將此值設得太高會降低 Amazon EC2 Auto Scaling 運作狀態檢查的有效性。如果您針對執行個體啟動使用 lifecycle hook，則可將運作狀態檢查寬限期的值設為 0。借助 lifecycle hook，Amazon EC2 Auto Scaling 提供了一種確保執行個體永遠會在進入 InService 狀態之前先初始化的方式。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling lifecycle hook](#)。

寬限期適用於以下執行個體：

- 新啟動的執行個體
- 待命後重新投入服務的執行個體
- 手動連接至群組的執行個體

Important

在運作狀態檢查寬限期內，如果 Amazon EC2 Auto Scaling 偵測到執行個體不再處於 Amazon EC2 running 狀態，則會立即將執行個體標示為 Unhealthy 並取代之。例如，如果您停止 Auto Scaling 群組中的執行個體，則會將該執行個體標示為 Unhealthy 並取代之。

設定群組的運作狀態檢查寬限期

您可以設定新的和現有 Auto Scaling 群組的運作狀態檢查寬限期。

Console

修改新群組的運作狀態檢查寬限期

當您建立 Auto Scaling 群組時，請在設定進階選項頁面、運作狀態檢查、運作狀態檢查寬限期中輸入時間量（以秒為單位）。這是 Amazon EC2 Auto Scaling 在進入 InService 狀態後，必須等待多久才能檢查執行個體的運作狀態。

AWS CLI

修改新群組的運作狀態檢查寬限期

將 `--health-check-grace-period` 選項新增至 [create-auto-scaling-group](#) 命令。以下範例針對名為 `my-asg` 的新 Auto Scaling 群組，設定運作狀態檢查寬限期，並將值設為 60 秒。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-grace-period 60 ...
```


Console

修改現有群組的運作狀態檢查寬限期

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所在的 AWS 區域。
3. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在 Details (詳細資訊) 索引標籤上，選擇 Health checks (運作狀態檢查)、Edit (編輯)。
5. 在 Health check grace period (運作狀態檢查寬限期) 之下，輸入時間量，以秒為單位。這是 Amazon EC2 Auto Scaling 在進入 InService 狀態後，必須等待多久才能檢查執行個體的運作狀態。
6. 選擇更新。

AWS CLI

修改現有群組的運作狀態檢查寬限期

將 `--health-check-grace-period` 選項新增至 [update-auto-scaling-group](#) 命令。以下範例針對名為 `my-asg` 的現有 Auto Scaling 群組，設定運作狀態檢查寬限期，並將值設為 `120` 秒。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-grace-period 120
```

Note

強烈建議還要設定 Auto Scaling 群組的預設執行個體預熱時間。如需詳細資訊，請參閱[設定 Auto Scaling 群組的預設執行個體暖機期](#)。

使用運作狀態檢查監控 Amazon EBS 磁碟區受損的 Auto Scaling 執行個體

您可以開啟 Auto Scaling 群組的 Amazon EBS 運作狀態檢查，以確保 Amazon EC2 Auto Scaling 監控應用程式執行所在的整個系統。

開啟這些運作狀態檢查後，Amazon EC2 Auto Scaling 會收到在執行個體連接的 EBS 磁碟區上執行的 Amazon EC2 狀態檢查結果。如果磁碟區無法連線或未通過 I/O 狀態檢查，則運作狀態檢查將會失敗，而對應的執行個體會被認定為運作狀態不佳。當 Amazon EC2 Auto Scaling 偵測到運作狀態不佳的執行個體時，會予以取代。

本主題假設您熟悉連接的 EBS 狀態檢查。如果不是，請參閱 Amazon EC2 使用者指南中的[連接 EBS 狀態檢查](#)一節，以取得詳細資訊。下列主題說明如何開啟依賴連接的 EBS 狀態檢查的 Amazon EC2 Auto Scaling 運作狀態檢查。

Note

您可以開啟所有 Auto Scaling 群組的 Amazon EBS 運作狀態檢查。不過，這些運作狀態檢查僅適用於[在 AWS Nitro 系統上建置的執行個體](#)。

開啟群組的 Amazon EBS 運作狀態檢查

您可以為新的和現有的 Auto Scaling 群組開啟 Amazon EBS 運作狀態檢查。

Console

開啟新群組的 Amazon EBS 運作狀態檢查

當您建立 Auto Scaling 群組時，在設定進階選項頁面上，針對運作狀態檢查、其他運作狀態檢查類型，選取開啟 Amazon EBS 運作狀態檢查。然後，在運作狀態檢查寬限期下，輸入時間量，以秒為單位。此時間量是 Amazon EC2 Auto Scaling 在進入 InService 狀態後，必須等待多久才能檢查執行個體的運作狀態。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。

AWS CLI

開啟新群組的 Amazon EBS 運作狀態檢查

將 `--health-check-type` 選項新增至 [create-auto-scaling-group](#) 命令。下列範例 EBS 會為名為的新 Auto Scaling 群組指定 `--health-check-type` 選項 `my-asg`。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60 ...
```

您可以為 `--health-check-type` 選項指定多個值。例如，若要同時新增 Amazon EBS 和 Elastic Load Balancing 運作狀態檢查類型，請使用下列命令。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
--health-check-type "EBS,ELB" --health-check-grace-period 60 ...
```

值名稱區分大小寫。

Console

開啟現有群組的 Amazon EBS 運作狀態檢查

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所在的 AWS 區域。
3. 選取現有群組旁邊的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

4. 在 Details (詳細資訊) 索引標籤上，選擇 Health checks (運作狀態檢查)、Edit (編輯)。
5. 針對運作狀態檢查、其他運作狀態檢查類型，選取開啟 Amazon EBS 運作狀態檢查。
6. 對於運作狀態檢查寬限期，輸入時間量，以秒為單位。此時間量是 Amazon EC2 Auto Scaling 在進入 InService 狀態後，必須等待多久才能檢查執行個體的運作狀態。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。
7. 選擇更新。

AWS CLI

開啟現有群組的 Amazon EBS 運作狀態檢查

將 --health-check-type 選項新增至 [update-auto-scaling-group](#) 命令。下列範例 EBS 會為名為的現有 Auto Scaling 群組指定 --health-check-type 選項 *my-asg*。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--health-check-type "EBS" --health-check-grace-period 60
```

若要使用多個運作狀態檢查類型，您可以為 --health-check-type 選項指定多個值（例如 **EBS,ELB**）。

值名稱區分大小寫。

關閉 Auto Scaling 群組的 Amazon EBS 運作狀態檢查

下列主題說明如何關閉 Auto Scaling 群組的 Amazon EBS 運作狀態檢查。如果您不再需要 Amazon EBS 運作狀態檢查，請使用下列程序將其關閉。

Console

關閉群組的 Amazon EBS 運作狀態檢查

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Health checks (運作狀態檢查)、Edit (編輯)。
4. 對於運作狀態檢查、其他運作狀態檢查類型，取消選取開啟 Amazon EBS 運作狀態檢查。
5. 選擇更新。

AWS CLI

關閉群組的 Amazon EBS 運作狀態檢查

若要更新 Auto Scaling 群組上的運作狀態檢查，使其不再使用 Amazon EBS 運作狀態檢查，請使用 [update-auto-scaling-group](#) 命令。包含 `--health-check-type` 選項和 **EC2** 的值。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--health-check-type "EC2"
```

若要關閉 Amazon EBS 運作狀態檢查而不關閉其他運作狀態檢查類型（例如 Elastic Load Balancing），您必須指定它們，而不是 **EC2**。例如，對於 Elastic Load Balancing 運作狀態檢查，請 **ELB** 為 `--health-check-type` 選項指定。

值名稱區分大小寫。

為您的 Auto Scaling 群組設定自訂運作狀態檢查

您可以使用自訂運作狀態檢查來補充 Amazon EC2 Auto Scaling 提供的現有運作狀態檢查選項。透過將自訂運作狀態檢查與其他運作狀態檢查類型結合，您可以建立針對應用程式需求量身打造的完整運作狀態監控系統。

若要開始使用，請建立自訂測試，以驗證 Auto Scaling 群組中的執行個體是否正常運作，並可處理傳入流量。如果您設定的運作狀態檢查偵測到執行個體沒有回應，請將該特定執行個體標記為 **Unhealthy**，這會導致 Amazon EC2 Auto Scaling 立即取代它。

您可以使用 `awscli` 或 SDK，將執行個體的運作狀態直接傳送至 AWS CLI Amazon EC2 Auto Scaling。下列範例示範如何使用 AWS CLI 來設定執行個體的運作狀態，然後驗證執行個體的運作狀態。

使用以下 [set-instance-health](#) 命令將指定的執行個體的運作狀態設定為 **Unhealthy**。

```
aws autoscaling set-instance-health --instance-id i-1234567890abcdef0 --health-status Unhealthy
```

根據預設，此指令遵循運作狀態檢查寬限期。但是，您可以覆寫這個行為，藉由納入 `--no-should-respect-grace-period` 選項而不遵守寬限期。

使用以下 [describe-auto-scaling-groups](#) 命令驗證該執行個體的運作狀態為 **Unhealthy**。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

以下是範例回應，顯示執行個體的運作狀態為 **Unhealthy**，且執行個體正在終止。

```
{
  "AutoScalingGroups": [
    {
      ....
      "Instances": [
        {
          "ProtectedFromScaleIn": false,
          "AvailabilityZone": "us-west-2a",
          "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1",
            "LaunchTemplateId": "lt-1234567890abcdef0"
          },
          "InstanceId": "i-1234567890abcdef0",
          "InstanceType": "t2.micro",
          "HealthStatus": "Unhealthy",
          "LifecycleState": "Terminating"
        },
        ...
      ]
    }
  ]
}
```

```
}  
]  
}
```

檢視運作狀態檢查失敗的原因

使用下列程序，您就能檢視因運作狀態檢查而被取代之任何執行個體的相關資訊。

Amazon EC2 Auto Scaling 預設會建立新的擴展活動來終止運作狀態不佳的執行個體，然後將其終止。在該執行個體終止時，另一個擴展活動會啟動新的執行個體。您可以使用執行個體維護政策，變更此行為以盡快開始啟動新的執行個體。如需詳細資訊，請參閱[執行個體維護政策](#)。

Console

檢視運作狀態檢查失敗的原因

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Activity (活動) 索引標籤的 Activity history (活動歷史記錄) 中，Status (狀態) 欄位會顯示 Auto Scaling 群組是否已成功啟動或終止執行個體。

若其已終止任何運作狀態不佳的執行個體，Cause (原因) 資料欄會顯示終止的日期和時間，以及運作狀態檢查失敗的原因。例如 At 2022-05-14T20:11:53Z an instance was taken out of service in response to a user health-check。此訊息表示自訂運作狀態檢查標示執行個體運作狀態不佳。

如需運作狀態檢查失敗的說明，請參閱 [對 Amazon EC2 Auto Scaling 中運作狀態不佳的執行個體進行故障診斷](#)。

AWS CLI

檢視運作狀態檢查失敗的原因

使用以下 [describe-scaling-activities](#) 命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下是回應範例，其中 Cause 包含運作狀態檢查失敗的原因。

```
{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in response to a user health-check.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:283179a2-f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
    ...
  ]
}
```

如需輸出中欄位的說明，請參閱《Amazon EC2 Auto Scaling API 參考》中的[活動](#)。

若要描述已刪除 Auto Scaling 群組之後的擴展活動，請將 `--include-deleted-groups` 選項新增至 [describe-scaling-activities](#) 命令。

對 Amazon EC2 Auto Scaling 中運作狀態不佳的執行個體進行故障診斷

以下是 Amazon EC2 Auto Scaling 傳回的錯誤訊息、潛在原因，以及您可以採取的步驟來解決問題。

若要擷取錯誤訊息，請參閱 [檢視運作狀態檢查失敗的原因](#)。

錯誤訊息

- [執行個體因 EC2 執行個體狀態檢查失敗而停止運作](#)
- [執行個體因指出了執行個體已終止或停止的 EC2 運作狀態檢查而停止運作](#)
- [執行個體因 ELB 系統運作狀態檢查失敗而停止運作](#)
- [其他資源](#)

執行個體因 EC2 執行個體狀態檢查失敗而停止運作

問題：Auto Scaling 執行個體未通過 Amazon EC2 狀態檢查。

原因 1：如果有問題導致 Amazon EC2 考慮 Auto Scaling 群組中的執行個體受損，Amazon EC2 Auto Scaling 會自動取代執行個體，作為其運作狀態檢查的一部分。

解決方案 1：當執行個體狀態檢查失敗時，您通常必須透過變更執行個體組態來自行解決問題，直到應用程式不再顯示任何問題為止。若要解決此問題，請遵循這些步驟：

1. 手動建立不屬於 Auto Scaling 群組的 Amazon EC2 執行個體，並調查問題。如需調查受損執行個體的一般說明，請參閱《Amazon EC2 使用者指南》中的[故障診斷狀態檢查失敗的執行個體](#)。
2. 確認您的執行個體已成功啟動且運作正常後，請將新的無錯誤執行個體組態部署到 Auto Scaling 群組。
3. 刪除已建立的執行個體，以免持續為您的 AWS 帳戶支付費用。

執行個體因指出了執行個體已終止或停止的 EC2 運作狀態檢查而停止運作

問題：已停止、重新啟動或終止的 Auto Scaling 執行個體被替換。

原因 1：使用者手動停止、重新啟動或終止執行個體。

解決方案 1：如果您需要停止或重新啟動 Auto Scaling 群組中的執行個體，建議您先讓執行個體處於待命狀態。如需詳細資訊，請參閱[從 Auto Scaling 群組暫時移除執行個體](#)。

原因 2：Amazon EC2 Auto Scaling 嘗試在 Amazon EC2 Spot 服務中斷執行個體後替換 Spot 執行個體，因為 Spot 價格增長超過您的最高價格或容量已不可使用。

解決方案 2：無法保證 Spot 執行個體在任何指定時間點都能滿足請求。不過，您可以嘗試下列方法：

- 使用較高的 Spot 最高價格 (可能是隨需價格)。將最高價格設得更高，可讓 Amazon EC2 Spot 服務更好地啟動和維護所需容量。
- 在多個可用區域中執行多個執行個體類型，增加可從中啟動執行個體的不同容量集區的數量。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。
- 如果使用多個執行個體類型，請考慮啟用容量重新平衡功能。如果希望 Amazon EC2 Spot 服務在執行中的執行個體終止前嘗試啟動新的 Spot 執行個體，則可用到此功能。如需詳細資訊，請參閱[Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體](#)。

原因 3：使用容量區塊時，Amazon EC2 會終止容量區塊結束時間前 30 分鐘仍在執行的任何執行個體。此突然終止會導致 Auto Scaling 群組嘗試啟動新執行個體以維持其所需的容量，即使容量區塊已結束。

解決方案 3：若要解決此問題，請嘗試下列動作：

- 減少 Auto Scaling 群組所需的容量，以防止其嘗試啟動新的執行個體。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 手動擴展](#)。
- 請務必在容量區塊結束時間前 30 分鐘擴展 Auto Scaling 群組，以免經常遇到此錯誤。確定任何生命週期關聯在容量區塊結束時間之前 30 分鐘完成。如需詳細資訊，請參閱[Capacity Blocks 用於機器學習工作負載](#)。

執行個體因 ELB 系統運作狀態檢查失敗而停止運作

問題：Auto Scaling 執行個體可能未通過 EC2 狀態檢查。但是，對於已註冊 Auto Scaling 群組的目標群組或 Classic Load Balancer，它們可能無法通過 Elastic Load Balancing 運作狀態檢查。

原因 1：如果您的 Auto Scaling 群組依賴 Elastic Load Balancing 提供的運作狀態檢查，Amazon EC2 Auto Scaling 會同時檢查 EC2 狀態檢查和 Elastic Load Balancing 運作狀態檢查的結果，以判斷執行個體的運作狀態。為了執行運作狀態檢查，負載平衡器會傳送請求至每個執行個體並等待正確的回應，或建立與執行個體的連線。執行個體可能無法通過 Elastic Load Balancing 運作狀態檢查，因為在執行個體上執行的應用程式問題造成負載平衡器認定執行個體已停止服務。

解決方案 1：若要通過 Elastic Load Balancing 運作狀態檢查：

- 確認目標群組的運作狀態檢查設定已正確設定。您可以依據目標群組，指定負載平衡器的運作狀態檢查設定。如需詳細資訊，請參閱[設定目標的運作狀態檢查](#)。
- 記下負載平衡器預期的成功代碼，並且確認您的應用程式已設定為在成功時傳回這些代碼。
- 確認負載平衡器和 Auto Scaling 群組的安全群組已正確設定。
- 確認負載平衡器設定在與 Auto Scaling 群組相同的可用區域中。

解決方案 2：更新 Auto Scaling 群組以關閉 Elastic Load Balancing 運作狀態檢查。如需如何關閉這些運作狀態檢查的指示，請參閱[分離目標群組或 Classic Load Balancer](#)。

原因 2：運作狀態檢查寬限期與執行個體啟動時間不相符。

解決方案 3：編輯 Auto Scaling 群組的運作狀態檢查寬限期。將寬限期設定為足夠長的時間，以支援 Elastic Load Balancing 認為新啟動的執行個體運作狀態良好之前所需的連續成功運作狀態檢查次數。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。

其他資源

如果您有不同的問題，請參閱下列 AWS re:Post 文章以取得其他故障診斷說明：

- [Amazon EC2 Auto Scaling 為什麼會終止執行個體？](#)
- [為什麼 Amazon EC2 Auto Scaling 未終止運作狀態不佳的執行個體？](#)

Health 儀板表 Amazon EC2 Auto Scaling 的通知

您的 Health 儀板表 支援來自 Amazon EC2 Auto Scaling 的通知。這些通知讓您了解可能影響您應用程式的資源效能或可用性問題，並提供解決之道。目前只有缺少安全群組和啟動範本的特定事件可供使用。

Health 儀板表 是 AWS Health 服務的一部分。其不需要設定，而且您帳戶中經過驗證的任何使用者皆可檢視。如需詳細資訊，請參閱 [AWS Health 儀表板入門](#)。

如果您收到的訊息類似以下訊息，應該將其視為警示，以便採取動作。

範例：由於缺少安全群組，Auto Scaling 群組並未擴增

Hello,

At 2020-01-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in AWS ## 123456789012.

A security group associated with this Auto Scaling group cannot be found. Each time a scale out operation is performed, it will be prevented until you make a change that fixes the issue.

We recommend that you review and update your Auto Scaling group configuration to change the launch template or launch configuration that depends on the unavailable security group.

Sincerely,
Amazon Web Services

範例：由於缺少啟動範本，Auto Scaling 群組並未擴增

Hello,

At 2021-05-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in AWS ## 123456789012.

The launch template associated with this Auto Scaling group cannot be found. Each time a scale out operation is performed, it will be prevented until you make a change that fixes the issue.

We recommend that you review and update your Auto Scaling group configuration and specify an existing launch template to use.

Sincerely,
Amazon Web Services

監控 Auto Scaling 群組和執行個體的 CloudWatch 指標

指標為 Amazon CloudWatch 中的基本概念。指標代表按時間順序發佈到 CloudWatch 的一組資料點。您可以將指標視為要監控的變數，且資料點代表該變數隨著時間的值。您可以使用這些指標來確認您的系統是否依照預期執行。

收集有關 Auto Scaling 群組資訊的 Amazon EC2 Auto Scaling 指標位於 AWS/AutoScaling 命名空間中。從 Auto Scaling 執行個體收集 CPU 和其他使用情況資料的 Amazon EC2 執行個體指標位於 AWS/EC2 命名空間中。

Amazon EC2 Auto Scaling 主控台會顯示群組指標的一系列圖形以及群組的彙總執行個體指標。根據需求，您可能偏好從 Amazon CloudWatch 而非 Amazon EC2 Auto Scaling 主控台中存取 Auto Scaling 群組和執行個體的資料。

如需詳細資訊，請參閱 [Amazon CloudWatch 使用者指南](#)。

目錄

- [在 Amazon EC2 Auto Scaling 主控台中檢視監控圖表](#)
- [Amazon EC2 Auto Scaling 的 Amazon CloudWatch 指標](#)
- [設定 Auto Scaling 執行個體的監控](#)

在 Amazon EC2 Auto Scaling 主控台中檢視監控圖表

在 Amazon EC2 主控台的 Amazon EC2 Auto Scaling 區段中，您可以使用 CloudWatch 指標監控個別 Auto Scaling 群組的逐分鐘進度。

您可以監控以下類型的指標：

- Auto Scaling 指標 – 僅當您啟用 Auto Scaling 指標時才會打開這些指標。如需詳細資訊，請參閱[啟用 Auto Scaling 群組指標 \(主控台\)](#)。啟用 Auto Scaling 指標後，監控圖表會顯示 Auto Scaling 指標以一分鐘粒度發佈的資料。
- EC2 指標 – 始終啟用 Amazon EC2 執行個體指標。如果已啟用詳細監控，則監控圖表會顯示每分鐘發佈的執行個體指標資料。如需詳細資訊，請參閱[設定 Auto Scaling 執行個體的監控](#)。

使用 Amazon EC2 Auto Scaling 主控台檢視監控圖表

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取您想要檢視指標之 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 選擇 Monitoring (監控) 索引標籤。

Amazon EC2 Auto Scaling 會顯示 Auto Scaling 指標的監控圖表。

4. 要檢視群組的彙總執行個體指標之圖表，請選擇 EC2。

圖表動作

- 將滑鼠暫留在資料點上以檢視特定時間 (UTC) 的資料快顯。
- 若要放大圖表，請選擇圖表右上角選單工具 (三個垂直點) 中的 Enlarge (放大)。或者，選擇圖表頂部的最大化圖示。
- 透過選取其中一個預先定義的時段值，調整圖表中所顯示資料的時段。如果放大圖表，您可以選擇 Custom (自訂) 來定義您自己的時段。
- 從選單工具中選擇 Refresh (重新整理) 以更新圖表中的資料。
- 將游標拖曳到圖表資料上以選取特定範圍。然後，您可以在選單工具中選擇 Apply time range (套用時間範圍)。
- 從選單工具中選擇 View logs (檢視日誌)，在 CloudWatch 主控台中檢視關聯的日誌串流 (如果有)。

- 若要在 CloudWatch 中檢視圖表，請從選單工具中選擇 View in metrics (在指標中檢視)。這會帶您進入該圖表的 CloudWatch 頁面。這樣，您可以檢視更多資訊或存取歷史資訊，以便更好地了解 Auto Scaling 群組在較長時間內的變化情況。

Auto Scaling 群組的圖表指標

建立 Auto Scaling 群組後，您可以開啟 Amazon EC2 Auto Scaling 主控台，在 Monitoring (監控) 索引標籤中檢視群組的監控圖表。

在 Auto Scaling 區段中，圖表指標包含以下指標。這些指標提供了可作為潛在問題之指標的度量，例如終止執行個體數量或擱置執行個體的數量。您可以在 [Amazon EC2 Auto Scaling 的 Amazon CloudWatch 指標](#) 中尋找這些指標的定義。

顯示名稱	CloudWatch 指標名稱
群組大小下限	GroupMinSize
群組大小上限	GroupMaxSize
所需容量	GroupDesiredCapacity
服務中的執行個體	GroupInServiceInstances
待定執行個體	GroupPendingInstances
待命執行個體	GroupStandbyInstances
正在終止的執行個體	GroupTerminatingInstances
執行個體總數	GroupTotalInstances

在 EC2 區段中，您可以根據 Amazon EC2 執行個體的關鍵效能指標找到以下圖表指標。這些 EC2 指標是群組中所有執行個體的指標彙總。您可以在 Amazon EC2 使用者指南的 [列出執行個體可用的 CloudWatch 指標](#) 中找到這些指標的定義。

顯示名稱	CloudWatch 指標名稱
CPU 使用率	CPUUtilization

顯示名稱	CloudWatch 指標名稱
磁碟讀取	DiskReadBytes
磁碟讀取操作	DiskReadOps
磁碟寫入	DiskWriteBytes
磁碟寫入操作	DiskWriteOps
網路傳入	NetworkIn
網路輸出	NetworkOut
狀態檢查失敗 (全部)	StatusCheckFailed
狀態檢查失敗 (執行個體)	StatusCheckFailed_Instance
狀態檢查失敗 (系統)	StatusCheckFailed_System

此外，某些指標適用於 Auto Scaling 圖表指標中的特定使用案例。

如果您的群組使用權重定義每個執行個體為群組的所需容量貢獻的單位數量，下列指標會相當實用。您可以在 [Amazon EC2 Auto Scaling 的 Amazon CloudWatch 指標](#) 中尋找這些指標的定義。

顯示名稱	CloudWatch 指標名稱
服務中容量單位	GroupInServiceCapacity
待定容量單位	GroupPendingCapacity
備用容量單位	GroupStandbyCapacity
終止容量單位	GroupTerminatingCapacity
總容量單位	GroupTotalCapacity

如果您的群組使用[暖集區](#)功能，下列指標會相當實用。您可以在 [Amazon EC2 Auto Scaling 的 Amazon CloudWatch 指標](#) 中尋找這些指標的定義。

顯示名稱	CloudWatch 指標名稱
暖集區大小下限	WarmPoolMinSize
暖集區所需容量	WarmPoolDesiredCapacity
暖集區擱置的容量單位	WarmPoolPendingCapacity
暖集區正在終止的容量單位	WarmPoolTerminatingCapacity
暖集區作用中容量單位	WarmPoolWarmedCapacity
暖集區已啟動的總容量單位	WarmPoolTotalCapacity
群組和暖集區所需容量	GroupAndWarmPoolDesiredCapacity
群組和暖集區已啟動的總容量單位	GroupAndWarmPoolTotalCapacity

相關資源

- 若要監控每個執行個體指標，請參閱《Amazon EC2 使用者指南》中的[為執行個體繪製指標](#)圖表。
- CloudWatch 儀表板是 CloudWatch 主控台中可自訂的首頁。您可以使用這些頁面，以在單一檢視中監控資源，甚至包括散佈在不同區域的資源。您可以使用 CloudWatch 儀表板，為您的 AWS 資源建立指標和警示的自訂檢視。如需更多資訊，請參閱[Amazon CloudWatch 使用者指南](#)。

Amazon EC2 Auto Scaling 的 Amazon CloudWatch 指標

Amazon EC2 Auto Scaling 在 AWS/AutoScaling 命名空間中發佈下列指標。可用的實際 Auto Scaling 群組指標將取決於您是否啟用群組指標，以及您已啟用的群組指標。群組指標可以以 1 分鐘的精細程度提供，且無需支付額外費用，但是您必須啟用這些指標。

啟用 Auto Scaling 群組後，Amazon EC2 Auto Scaling 會竭盡全力每分鐘將取樣資料傳送到 CloudWatch。在 CloudWatch 遇到服務中斷的極少數情況下，資料不會回填以填補群組指標歷史記錄中的空白。

目錄

- [Auto Scaling 群組指標](#)

- [Auto Scaling 群組指標維度](#)
- [預測擴展指標與維度](#)
- [啟用 Auto Scaling 群組指標 \(主控台\)](#)
- [啟用 Auto Scaling 群組指標 \(AWS CLI\)](#)

Auto Scaling 群組指標

使用這些指標，您可以持續地了解 Auto Scaling 群組歷史記錄，例如群組大小隨時間的變化。

指標	Description
GroupMinSize	Auto Scaling 群組的最低大小。 報告條件：如果啟用指標收集，則會報告。
GroupMaxSize	Auto Scaling 群組的最高大小。 報告條件：如果啟用指標收集，則會報告。
GroupDesiredCapacity	Auto Scaling 群組嘗試維持的執行個體數量。 報告條件：如果啟用指標收集，則會報告。
GroupInServiceInstances	執行為 Auto Scaling 群組的一部分的執行個體數量。此指標不包含待定或正在終止的執行個體。 報告條件：如果啟用指標收集，則會報告。
GroupPendingInstances	待定的執行個體的數量。待定的執行個體尚未提供服務。此指標不包含服務中或終止的執行個體。 報告條件：如果啟用指標收集，則會報告。
GroupStandbyInstances	處於 Standby 狀態的執行個體的數量。此狀態的執行個體仍持續執行，但不主動提供服務。 報告條件：如果啟用指標收集，則會報告。

指標	Description
GroupTerminatingInstances	正在執行終止程序的執行個體的數量。此指標不包括 Auto Scaling 群組縮減後處於服務中、擱置中或返回暖集區的執行個體。 報告條件：如果啟用指標收集，則會報告。
GroupTerminatingRetainedInstances	狀態中的執行個體數目 Terminating:Retained 。 報告條件：如果啟用指標收集，則會報告。
GroupTotalInstances	Auto Scaling 群組中的執行個體總數。此指標識別服務中、待定及終止的執行個體的數量。 報告條件：如果啟用指標收集，則會報告。

當您設定混合執行個體群組以不同單位測量其所需容量時 (例如根據每個執行個體類型的 vCPU 數指派權重)，下列指標會計入 Auto Scaling 群組使用的單位數量。如果未設定混合執行個體群組以不同單位來測量其所需容量，則會填入下列指標，但是會等於上一個表格中定義的指標。如需詳細資訊，請參閱[建立混合執行個體群組的設定概觀](#)。

指標	Description
GroupInServiceCapacity	作為 Auto Scaling 群組一部分執行的容量單位數目。 報告條件：如果啟用指標收集，則會報告。
GroupPendingCapacity	擱置中的容量單位數目。 報告條件：如果啟用指標收集，則會報告。
GroupStandbyCapacity	處於 Standby 狀態的容量單位數目。 報告條件：如果啟用指標收集，則會報告。
GroupTerminatingCapacity	正在終止處理的容量單位數目。 報告條件：如果啟用指標收集，則會報告。

指標	Description
GroupTerminatingRetainedCapacity	狀態中的執行個體所代表的容量單位Terminating:Retained 。 報告條件：如果啟用指標收集，則會報告。
GroupTotalCapacity	Auto Scaling 群組中的容量單位總數。 報告條件：如果啟用指標收集，則會報告。

Amazon EC2 Auto Scaling 也會針對具有暖集區的 Auto Scaling 群組報告以下指標。如需詳細資訊，請參閱[使用暖集區縮短開機時間的應用程式延遲](#)。

指標	Description
WarmPoolMinSize	暖集區的最小尺寸。 報告條件：如果啟用指標收集，則會報告。
WarmPoolDesiredCapacity	Amazon EC2 Auto Scaling 嘗試在暖集區中維護的容量。 這相當於 Auto Scaling 群組的最大大小減去其所需容量，如果設定，則相當於 Auto Scaling 組的最大準備容量減去所需容量。 但是，如果暖集區的最小大小等於或大於 Auto Scaling 群組的最大大小 (或是已設定之最大準備容量) 與所需容量之間的差值，則暖集區所需容量會等同於 WarmPoolMinSize 。 報告條件：如果啟用指標收集，則會報告。
WarmPoolPendingCapacity	擱置暖集區中的容量。這包括 Auto Scaling 群組縮減後返回暖集區的執行個體。此指標不包含執行中、已停用或正在終止的執行個體。 報告條件：如果啟用指標收集，則會報告。
WarmPoolPendingRetainedCapacity	追蹤返回暖集區的保留執行個體。

指標	Description
	報告條件：如果啟用指標收集，則會報告。
WarmPoolTerminatingCapacity	正在終止的暖集區中的容量。此指標不包含執行中、已停用或擱置的執行個體。 報告條件：如果啟用指標收集，則會報告。
WarmPoolTerminatingRetainedCapacity	追蹤從暖集區終止的保留執行個體。 報告條件：如果啟用指標收集，則會報告。
WarmPoolWarmedCapacity	在水平擴展期間進入 Auto Scaling 群組的可用容量。此指標不包含待定或正在終止的執行個體。 報告條件：如果啟用指標收集，則會報告。
WarmPoolTotalCapacity	暖集區的總容量，包含執行中、已停用、擱置或正在終止的執行個體。 報告條件：如果啟用指標收集，則會報告。
GroupAndWarmPoolDesiredCapacity	Auto Scaling 群組和暖集區組合所需的容量。 報告條件：如果啟用指標收集，則會報告。
GroupAndWarmPoolTotalCapacity	Auto Scaling 群組和暖集區組合的總容量。其中包含執行中、已停用、擱置、正在終止或正在使用的執行個體。 報告條件：如果啟用指標收集，則會報告。

Auto Scaling 群組指標維度

您可以使用下列維度來強化上表中列出的指標。

維度	Description
AutoScalingGroupName	對 Auto Scaling 群組的名稱進行篩選。

預測擴展指標與維度

AWS/AutoScaling 命名空間包含下列預測擴展指標。

指標的解析度為一小時。

您可以透過比較預測值與實際值，來評估預測準確性。如需有關評估預測準確性的詳細資訊，請參閱 [使用 CloudWatch 監控預測擴展指標](#)。

指標	Description	維度
PredictiveScalingLoadForecast	您的應用程式預期會產生的負載量。 Average、Minimum 以及 Maximum 統計資料有用，但 Sum 統計資料無用。 報告條件：在建立初始預測之後報告。	AutoScalingGroupName , PolicyName , PairIndex
PredictiveScalingCapacityForecast	滿足應用程式需求所需的預期容量。這以您要維護 Auto Scaling 執行個體所需的負載預測和目標使用率層級為依據。 Average、Minimum 以及 Maximum 統計資料有用，但 Sum 統計資料無用。 報告條件：在建立初始預測之後報告。	AutoScalingGroupName , PolicyName
PredictiveScalingMetricPairCorrelation	擴展指標與負載指標的每個執行個體平均值之間的相互關聯性。預測擴展會假設高相互關聯性。因此，如果您在此指標中觀察到偏低的值，最好不要使用指標對。 Average、Minimum 以及 Maximum 統計資料有用，但 Sum 統計資料無用。 報告條件：在建立初始預測之後報告。	AutoScalingGroupName , PolicyName , PairIndex

 Note

PairIndex 維度會傳回與 Amazon EC2 Auto Scaling 指派之負載擴展指標對索引相關聯的資訊。目前唯一有效的值為：0。

啟用 Auto Scaling 群組指標 (主控台)

啟用群組指標

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在 Monitoring (監控) 索引標籤上，選取 Auto Scaling group metrics collection (Auto Scaling 群組指標集合)，再在 Auto Scaling 下選擇 Enable (啟用) 位於頁面頂端的核取方塊。

停用群組指標

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組。
3. 在 Monitoring (監控) 索引標籤上，清除 Auto Scaling group metrics collection (Auto Scaling 群組指標集合)，再選擇 Enable (啟用) 核取方塊。

啟用 Auto Scaling 群組指標 (AWS CLI)

啟用 Auto Scaling 群組指標

使用 [enable-metrics-collection](#) 命令，啟用一個或多個群組指標。例如，以下命令可針對特定的 Auto Scaling 群組啟用單一指標。

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \  
--metrics GroupDesiredCapacity --granularity "1Minute"
```

如果您省略 `--metrics` 選項，則會啟用所有指標。


```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \  
--granularity "1Minute"
```

停用 Auto Scaling 群組指標

使用 [disable-metrics-collection](#) 命令來停用所有群組指標。

```
aws autoscaling disable-metrics-collection --auto-scaling-group-name my-asg
```

設定 Auto Scaling 執行個體的監控

Amazon EC2 會收集來自執行個體的原始資料，並處理為可讀且近乎即時的指標，它們會說明 Auto Scaling 群組的 CPU 和其他用量資料。透過選擇一分鐘或五分鐘的間隔長度，可設定監控這些指標的間隔。

每當執行個體啟動時，就會啟用執行個體監控功能，使用基本監控 (五分鐘間隔長度) 或詳細監控 (一分鐘間隔長度)。如需詳細監控，需額外收費。如需詳細資訊，請參閱 [《Amazon EC2 使用者指南》中的 Amazon CloudWatch 定價](#) 和 [使用 CloudWatch 監控執行個體](#)。Amazon EC2

建立 Auto Scaling 群組之前，您應該建立啟動範本或啟動組態，允許適用於您應用程式的監控類型。如果您將擴展政策新增到群組，強烈建議您使用詳細監控，以一分鐘的間隔長度取得 EC2 執行個體的指標資料，確保對負載變化作出更快回應。

目錄

- [啟用詳細監控 \(主控台\)](#)
- [啟用詳細監控 \(AWS CLI\)](#)
- [在基本監控和詳細監控之間切換](#)
- [使用 CloudWatch 代理程式收集其他指標](#)

啟用詳細監控 (主控台)

根據預設，當您使用 AWS 管理主控台 建立啟動範本或啟動組態時，會啟用基本監控。

在啟動範本中啟用詳細監控

當您使用 建立啟動範本時 AWS 管理主控台，在進階詳細資訊區段中，針對詳細 CloudWatch 監控，選擇啟用。否則會啟用基本監控。如需詳細資訊，請參閱[使用進階設定建立啟動範本](#)。

在啟動組態中啟用詳細監控

當您使用 建立啟動組態時 AWS 管理主控台，請在其他組態區段中，選取在 CloudWatch 中啟用 EC2 執行個體詳細監控。否則會啟用基本監控。如需詳細資訊，請參閱[建立啟動組態](#)。

啟用詳細監控 (AWS CLI)

根據預設，當您使用 AWS CLI 來建立啟動範本時，就會啟用基本監控。當您使用 AWS CLI 來建立啟動組態，依預設會啟用詳細監控。

在啟動範本中啟用詳細監控

對於啟動範本，請使用 [create-launch-template](#) 命令並傳遞 JSON 檔案，檔案中應包含用於建立啟動範本的資訊。將監控屬性設定為 "Monitoring":{"Enabled":true} 以啟用詳細監控，或設定為 "Monitoring":{"Enabled":false} 以啟用基本監控。

在啟動組態中啟用詳細監控

針對啟動組態，使用 [create-launch-configuration](#) 命令和 --instance-monitoring 選項。設定此選項為 true 以啟用詳細監控，或 false 只啟用基本監控。

```
--instance-monitoring Enabled=true
```

在基本監控和詳細監控之間切換

若要變更在新的 EC2 執行個體上啟用的監控類型，請更新啟動範本或更新 Auto Scaling 群組以使用新的啟動範本或啟動組態。現有執行個體會繼續使用先前啟用的監控類型。若要更新所有執行個體，請終止這些執行個體，以便由 Auto Scaling 群組取代，或使用 [monitor-instances](#) 和 [unmonitor-instances](#) 更新單個執行個體。

Note

使用執行個體重新整理和執行個體生命週期上限功能，您也可以取代 Auto Scaling 群組中所有的執行個體，以啟動使用新設定的新執行個體。如需詳細資訊，請參閱[取代 Auto Scaling 群組中的執行個體](#)。

在基本監控和詳細監控之間切換時：

如果您有與 Auto Scaling 群組的步進擴展政策或簡易擴展政策關聯的 CloudWatch 警示，請使用 [put-metric-alarm](#) 命令來更新每個警示。讓每個期間符合監控類型 (基本監控為 300 秒，而詳細監控為 60 秒)。如果您從詳細監控變更為基本監控，但並未更新您的警示以符合五分鐘的期間，則這些監控會持續每分鐘檢查統計資料。它們可能會發現每五個期間中有四個沒有可用資料。

使用 CloudWatch 代理程式收集其他指標

若要收集可用和已用記憶體等作業系統層級指標，您必須安裝 CloudWatch 代理程式。可能需支付額外費用。您可以使用 CloudWatch 代理程式，從 Amazon EC2 執行個體收集系統指標和日誌檔案。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [CloudWatch 代理程式收集的指標](#)。

使用 記錄 Amazon EC2 Auto Scaling API 呼叫 AWS CloudTrail

Amazon EC2 Auto Scaling 已與 整合 [AWS CloudTrail](#)，這項服務可提供使用者、角色或所採取動作的記錄 AWS 服務。CloudTrail 會將 Auto Scaling 的 API 呼叫擷取為事件。擷取的呼叫包括來自的 AWS 管理主控台 呼叫，以及對 Auto Scaling API 操作的程式碼呼叫。您可以使用 CloudTrail 所收集的資訊，判斷對 Auto Scaling 提出的請求、提出請求的 IP 地址、提出請求的時間，以及其他詳細資訊。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根使用者還是使用者憑證提出。
- 請求是否代表 IAM Identity Center 使用者提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務服務提出。

當您建立帳戶 AWS 帳戶 時 CloudTrail 會在 中處於作用中狀態，而且您會自動存取 CloudTrail 事件歷史記錄。CloudTrail 事件歷史記錄為 AWS 區域中過去 90 天記錄的管理事件，提供可檢視、可搜尋、可下載且不可變的記錄。如需詳細資訊，請參閱「AWS CloudTrail 使用者指南」中的 [使用 CloudTrail 事件歷史記錄](#)。檢視事件歷史記錄不會產生 CloudTrail 費用。

如需 AWS 帳戶 過去 90 天內持續記錄的事件，請建立追蹤。

CloudTrail 追蹤

線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。使用 建立的所有線索 AWS 管理主控台 都是多區域。您可以使用 AWS CLI 建立單一或多區域追蹤。建議您建立多區域追蹤，因為您擷取 AWS 區域 帳戶中所有的活動。如果您建立單一區域追蹤，您只能檢視追蹤 AWS 區域中記錄的事件。如需追蹤的詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的 [為您的 AWS 帳戶建立追蹤](#) 和 [為組織建立追蹤](#)。

您可以透過建立追蹤，免費將持續管理事件的一個複本從 CloudTrail 傳遞至您的 Amazon S3 儲存貯體，但這樣做會產生 Amazon S3 儲存費用。如需 CloudTrail 定價的詳細資訊，請參閱 [AWS CloudTrail 定價](#)。如需 Amazon S3 定價的相關資訊，請參閱 [Amazon S3 定價](#)。

CloudTrail 中的 Auto Scaling 管理事件

[管理事件](#)提供在資源上執行的管理操作的相關資訊 AWS 帳戶。這些也稱為控制平面操作。根據預設，CloudTrail 記錄管理事件。

Amazon EC2 Auto Scaling 會將所有 Auto Scaling 控制平面操作記錄為管理事件。如需 Auto Scaling 記錄到 CloudTrail 的 Amazon EC2 Auto Scaling 控制平面操作清單，請參閱 [Amazon EC2 Auto Scaling API 參考](#)。

Auto Scaling 事件範例

一個事件代表任何來源提出的單一請求，並包含請求 API 操作的相關資訊、操作的日期和時間、請求參數等。CloudTrail 日誌檔案不是公有 API 呼叫的已排序堆疊追蹤，因此事件不會以任何特定順序顯示。

以下範例顯示的 CloudTrail 事件會示範 CreateLaunchConfiguration 操作。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "Root",
    "principalId": "123456789012",
    "arn": "arn:aws:iam::123456789012:root",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2018-08-21T17:05:42Z"
      }
    }
  },
  "eventTime": "2018-08-21T17:07:49Z",
  "eventSource": "autoscaling.amazonaws.com",
  "eventName": "CreateLaunchConfiguration",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "Coral/Jakarta",
  "requestParameters": {
    "ebsOptimized": false,
    "instanceMonitoring": {
      "enabled": false
    }
  }
}
```

```

    },
    "instanceType": "t2.micro",
    "keyName": "EC2-key-pair-oregon",
    "blockDeviceMappings": [
      {
        "deviceName": "/dev/xvda",
        "ebs": {
          "deleteOnTermination": true,
          "volumeSize": 8,
          "snapshotId": "snap-01676e0a2c3c7de9e",
          "volumeType": "gp2"
        }
      }
    ],
    "launchConfigurationName": "launch_configuration_1",
    "imageId": "ami-6cd6f714d79675a5",
    "securityGroups": [
      "sg-00c429965fd921483"
    ]
  },
  "responseElements": null,
  "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
  "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}

```

如需有關 CloudTrail 記錄內容的資訊，請參閱《AWS CloudTrail 使用者指南》中的 [CloudTrail record contents](#)。

CloudWatch 上的 Auto Scaling RemoveAction 呼叫

當 Auto Scaling 指示 CloudWatch 從警示中移除自動擴展動作時，您的 AWS CloudTrail 日誌可能會顯示 Auto Scaling 呼叫 CloudWatch RemoveAction API。如果您取消註冊可擴展的目標、刪除擴展政策，或警示調用不存在的擴展政策，就可能會發生這種情況。

Amazon EC2 Auto Scaling 的 Amazon SNS 通知選項 Amazon EC2 Auto Scaling

您可以設定 Auto Scaling 群組，以通知您影響應用程式的重要事件。透過通知，您也可以消除輪詢，而且不會遇到有時由輪詢引起的 RequestLimitExceeded 錯誤。

有兩種方式可以接收有關 Amazon EC2 Auto Scaling 的通知：

- Amazon Simple Notification Service – Amazon SNS 可以在 Auto Scaling 群組啟動或終止執行個體時通知您。您只能開啟或關閉 Amazon SNS 通知。如需詳細資訊，請參閱[Amazon SNS 和 Amazon EC2 Auto Scaling](#)。
- Amazon EventBridge – EventBridge 提供更進階、符合指定條件的事件驅動通知，並傳送至各種目標，包括 Amazon SNS。EventBridge 也可以監控更廣泛的 Auto Scaling 事件，以進行更精確的監控。如需詳細資訊，請參閱[使用 EventBridge 處理 Auto Scaling 事件](#)。

您可以選擇使用具有生命週期掛鉤的通知，在啟動或終止期間對執行個體執行自訂動作。如需如何設定通知以搭配生命週期掛鉤使用的詳細資訊，請參閱[Amazon EC2 Auto Scaling lifecycle hook](#)。

Amazon SNS 和 Amazon EC2 Auto Scaling

本節說明如何使用 Amazon SNS 來監控 Auto Scaling 群組何時啟動或終止執行個體。

例如，如果您設定 Auto Scaling 群組使用 `autoscaling: EC2_INSTANCE_TERMINATE` 通知類型，當 Auto Scaling 群組終止執行個體時，它會傳送電子郵件通知。此電子郵件包含終止執行個體的詳細資訊，如執行個體的 ID 和終止執行個體的原因。

請注意，當 Amazon EC2 Auto Scaling 從群組新增或移除執行個體時，這些變更的通知會傳送給您，每個執行個體都會傳送一個通知。不過，這些通知的交付是盡最大努力的，您的執行個體在初始通知之後仍可能會失敗，例如，如果稍後的運作狀態檢查失敗。如需運作狀態檢查程序的詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

如需 Amazon SNS 一般的詳細資訊，請參閱[Amazon Simple Notification Service 開發人員指南](#)。

目錄

- [SNS 通知](#)
- [設定 Amazon EC2 Auto Scaling 的 Amazon SNS 通知](#)
 - [建立 Amazon SNS 主題](#)
 - [訂閱 Amazon SNS 主題](#)
 - [確認 Amazon SNS 訂閱](#)
 - [設定 Auto Scaling 群組以傳送通知](#)
 - [測試通知](#)
 - [刪除通知組態](#)
- [加密 Amazon SNS 主題的金鑰政策](#)

SNS 通知

Amazon EC2 Auto Scaling 支援在以下事件發生時傳送 Amazon SNS 通知。

事件	Description
autoscaling:EC2_INSTANCE_LAUNCH	執行個體成功啟動
autoscaling:EC2_INSTANCE_LAUNCH_ERROR	執行個體啟動失敗
autoscaling:EC2_INSTANCE_TERMINATE	執行個體成功終止
autoscaling:EC2_INSTANCE_TERMINATE_ERROR	執行個體終止失敗

訊息包括下列資訊：

- Event：事件。
- AccountId：Amazon Web Services 帳戶 ID。
- AutoScalingGroupName：Auto Scaling 群組的名稱。
- AutoScalingGroupARN：Auto Scaling 群組的 ARN。
- EC2InstanceId：EC2 執行個體的 ID。

例如：

```
Service: AWS Auto Scaling
Time: 2016-09-30T19:00:36.414Z
RequestId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:region:123456789012:autoScalingGroup...
ActivityId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Description: Launching a new EC2 instance: i-0598c7d356eba48d7
Cause: At 2016-09-30T18:59:38Z a user request update of AutoScalingGroup constraints to ...
```



```
StartTime: 2016-09-30T19:00:04.445Z
EndTime: 2016-09-30T19:00:36.414Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-0598c7d356eba48d7
Details: {"Subnet ID":"subnet-id","Availability Zone":"zone"}
Origin: AutoScalingGroup
Destination: EC2
```

設定 Amazon EC2 Auto Scaling 的 Amazon SNS 通知

若要使用 Amazon SNS 傳送電子郵件通知，您必須先建立主題，然後以您的電子郵件地址訂閱此主題。

建立 Amazon SNS 主題

SNS 主題是一種邏輯存取點，也是 Auto Scaling 群組用來傳送通知的通訊管道。您可以為您所建立的主題指定名稱。

在建立主題名稱時，名稱必須符合下列要求：

- 長度介於 1 與 256 個字元之間
- 包含大小寫 ASCII 字母、數字、底線或連字號

如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的[建立 Amazon SNS 主題](#)。

訂閱 Amazon SNS 主題

若要收到 Auto Scaling 群組傳送到該主題的通知，您必須訂閱連結至主題的端點。在此過程中，請於 Endpoint (端點) 指定要用來接收 Amazon EC2 Auto Scaling 通知的電子郵件地址。

如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的[訂閱 Amazon SNS 主題](#)。

確認 Amazon SNS 訂閱

Amazon SNS 會將確認電子郵件傳送到您在先前步驟中所指定的電子郵件地址。

請務必從 AWS 通知開啟電子郵件，並選擇連結確認訂閱後，再繼續進行下一個步驟。

您會收到來自 AWS 的確認訊息。Amazon SNS 現在已設定為採用您指定的電子郵件地址，以電子郵件方式接收和傳送通知。

設定 Auto Scaling 群組以傳送通知

您可以在啟動或終止執行個體等擴展事件發生時，設定 Auto Scaling 群組傳送通知給 Amazon SNS。Amazon SNS 會傳送執行個體的資訊通知到您所指定的電子郵件地址。

設定 Auto Scaling 群組的 Amazon SNS 通知 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格，顯示所選群組的相關資訊。

3. 在 Activity (活動) 索引標籤中，選擇 Activity notifications (活動通知)、Create notification (建立通知)。
4. 在 Create notifications (建立通知) 窗格，請執行下列動作：
 - a. 對於 SNS Topic (SNS 主題)，選取您的 SNS 主題。
 - b. 對於 Event types (事件類型)，選取要傳送通知的事件。
 - c. 選擇建立。

若要設定 Auto Scaling 群組的 Amazon SNS 通知 (AWS CLI)

請使用以下 [put-notification-configuration](#) 命令。

```
aws autoscaling put-notification-configuration --auto-scaling-group-name my-asg --topic-arn arn --notification-types "autoscaling:EC2_INSTANCE_LAUNCH" "autoscaling:EC2_INSTANCE_TERMINATE"
```

測試通知

若要產生啟動事件的通知，請將 Auto Scaling 群組所需的容量增加 1 來更新 Auto Scaling 群組。在執行個體啟動後的幾分鐘內，您會接收到通知。

變更所需的容量 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。

2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格，顯示所選群組的相關資訊。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Group details (群組詳細資訊)、Edit (編輯)。
4. 對於 Desired capacity (所需容量)，將目前值增加 1。如果此值超過 Maximum capacity (容量上限)，您必須也將 Maximum capacity (容量上限) 的值增加 1。
5. 選擇更新。
6. 幾分鐘之後，您會收到有關事件的通知。如果在此測試中，啟動了不需要的多餘執行個體，可以將 Desired capacity (所需容量) 減少 1。幾分鐘之後，您會收到有關事件的通知。

刪除通知組態

如果不再使用 Amazon EC2 Auto Scaling 通知組態，則可將其刪除。

刪除 Amazon EC2 Auto Scaling 通知組態 (主控台)

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組。
3. 在 Activity (活動) 索引標籤上，選取要刪除的通知旁的核取方塊，然後選擇 Actions (動作)、Delete (刪除)。

刪除 Amazon EC2 Auto Scaling 通知組態 (AWS CLI)

使用下列 delete-notification-configuration 命令。

```
aws autoscaling delete-notification-configuration --auto-scaling-group-name my-asg --  
topic-arn arn
```

如需刪除 Amazon SNS 主題以及所有與 Auto Scaling 群組相關聯訂閱的相關資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的[刪除 Amazon SNS 訂閱和主題](#)。

加密 Amazon SNS 主題的金鑰政策

您指定的 Amazon SNS 主題可能會使用透過 AWS Key Management Service 建立的客戶受管金鑰進行加密。若要授予 Amazon EC2 Auto Scaling 發佈到加密主題的許可，您必須先建立 KMS 金鑰，然

後將下列陳述式新增至 KMS 金鑰的政策。請將範例 ARN 取代為允許存取金鑰的適當服務連結角色之 ARN。如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的[設定 AWS KMS 許可](#)。

在此範例中，政策陳述式會向名為 `AWSServiceRoleForAutoScaling` 的服務連結角色授予許可，使其能夠使用客戶受管金鑰。若要進一步了解 Amazon EC2 Auto Scaling 服務連結角色的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 的服務連結角色](#)。

```
{
  "Sid": "Allow service-linked role use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

允許 Amazon EC2 Auto Scaling 發佈到加密主題的金鑰政策中不支援 `aws:SourceArn` 和 `aws:SourceAccount` 條件金鑰。

AWS 與 Amazon EC2 Auto Scaling 整合的 服務

Amazon EC2 Auto Scaling 可以與其他 AWS 服務整合。檢閱以下整合選項，進一步了解如何結合使用每項服務與 Amazon EC2 Auto Scaling。

主題

- [Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體](#)
- [使用容量預留在特定可用區域中預留容量](#)
- [使用 從命令列建立 Auto Scaling 群組 AWS CloudShell](#)
- [使用 建立 Auto Scaling 群組 AWS CloudFormation](#)
- [使用 取得執行個體類型建議 AWS Compute Optimizer](#)
- [使用 Elastic Load Balancing 在 Auto Scaling 群組中分配傳入應用程式流量](#)
- [使用 VPC Lattice 目標群組管理流量流程](#)
- [使用 EventBridge 處理 Auto Scaling 事件](#)
- [使用 Amazon VPC 為您的 Auto Scaling 執行個體提供網際網路連線](#)

Auto Scaling 中的容量重新平衡，以取代具有風險的 Spot 執行個體

Auto Scaling 中的容量重新平衡可協助您主動取代具有中斷風險的 Spot 執行個體，以維持工作負載可用性。

當 Spot 執行個體的中斷風險提高時，Amazon EC2 Spot 服務會將 EC2 執行個體重新平衡建議傳送至 Amazon EC2 Auto Scaling。如果您啟用容量重新平衡，Auto Scaling 會嘗試主動取代群組中已收到 EC2 執行個體重新平衡建議的 Spot 執行個體。這樣一來，您即可有機會將工作負載重新平衡至未處於提高之中斷風險的新的 Spot 執行個體。

當您不使用容量重新平衡時，在 Amazon EC2 Spot 服務中斷執行個體且其運作狀態檢查失敗之前，Auto Scaling 不會取代 Spot 執行個體。在中斷執行個體之前，Amazon EC2 必定會提供 EC2 執行個體重新平衡建議和 Spot 執行個體兩分鐘中斷通知。

目錄

- [概要](#)
- [容量重新平衡行為](#)

- [考量事項](#)
- [啟用容量重新平衡以主動取代具有風險的 Spot 執行個體](#)

概要

若要將容量重新平衡與 Auto Scaling 群組搭配使用，基本步驟如下：

1. 設定您的 Auto Scaling 群組，以使用多個執行個體類型和可用區域。這樣一來，Amazon EC2 Auto Scaling 就可以查看每個可用區域中 Spot 執行個體的可用容量。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。
2. 視需要新增生命週期關聯，以便在接收重新平衡通知的執行個體內對應用程式執行正常關閉程序。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling lifecycle hook](#)。

以下是您可能使用 lifecycle hook 的一些原因：

- 實現 Amazon SQS 工作者的順利關機
 - 完成從網域名稱系統 (DNS) 取消註冊
 - 提取系統或應用程式日誌並上傳到 Amazon Simple Storage Service (Amazon S3)
3. 開發 lifecycle hook 的自訂動作。若要盡快調用您的自訂動作，您需要知道執行個體何時可以終止。可以透過偵測執行個體的生命週期狀態來找出答案。
 - 若要調用執行個體以外的動作，請撰寫 EventBridge 規則，以在事件模式符合規則時自動化要採取的動作。
 - 若要在執行個體內調用動作，請將執行個體設定為執行關閉指令碼，並透過執行個體中繼資料擷取生命週期狀態。

設計自訂動作在兩分鐘內完成操作至關重要。這樣可以確保在執行個體終止前有充裕的時間完成任務。

完成這些步驟後，您便可以開始使用容量重新平衡。

容量重新平衡行為

透過容量重新平衡，當執行個體接收重新平衡建議時，Amazon EC2 Auto Scaling 的行為如下：

- 新 Spot 執行個體啟動時，Amazon EC2 Auto Scaling 會先等待新執行個體通過運作狀態檢查，然後才會終止先前的執行個體。在取代多個執行個體時，每個先前的執行個體都會在新執行個體啟動並通過其運作狀態檢查之後開始終止。

- 由於 Amazon EC2 Auto Scaling 會在終止先前的執行個體前嘗試啟動新執行個體，達到或接近最大指定容量便會阻礙或完全停止重新平衡的活動。若要避免這個問題，Amazon EC2 Auto Scaling 可以暫時超過該群組的大小上限最高可達所需容量的 10%。
- 如果您沒有將 lifecycle hook 新增到 Auto Scaling 群組，Amazon EC2 Auto Scaling 便會在新執行個體通過運作狀態檢查後立即終止先前的執行個體。
- 如果您新增了 lifecycle hook，便可藉由您為 lifecycle hook 指定的逾時值，來延長我們開始終止先前執行個體之前所需的時間。
- 如果您使用擴展政策或排程擴展，則擴展活動會並行執行。如果正在執行擴展活動，且 Auto Scaling 群組低於其所需的新容量，Amazon EC2 Auto Scaling 會在終止先前的執行個體之前進行擴增。

如果在一個可用區域中沒有執行個體類型的容量，Amazon EC2 Auto Scaling 會繼續嘗試在其他啟用的可用區域中啟動 Spot 執行個體，直到啟動成功為止。

在最壞的情況下，如果新執行個體無法啟動或其運作狀態檢查失敗，Amazon EC2 Auto Scaling 會繼續嘗試重新啟動這些執行個體。在其嘗試啟動新的執行個體時，會在發出兩分鐘的中斷通知後，最終中斷並強制終止先前的執行個體。

考量事項

使用容量重新平衡時，請考慮下列事項：

應用程式的設計應容許 Spot 中斷

您的應用程式應該能夠處理執行個體數量的動態變更以及 Spot 執行個體提前中斷的可能性。例如，如果 Auto Scaling 群組位於 Elastic Load Balancing 負載平衡器後方，Amazon EC2 Auto Scaling 會等待執行個體從負載平衡器取消註冊，然後再呼叫 lifecycle hook。如果取消註冊執行個體和完成生命週期動作的時間過長，在 Amazon EC2 Auto Scaling 等待您的生命週期動作完成後再終止執行個體期間，執行個體可能會中斷。

Amazon EC2 並不總是能在兩分鐘 Spot 執行個體中斷通知之前傳送重新平衡建議訊號。有時候，重新平衡建議訊號會與兩分鐘的中斷通知一起到達。發生這種情況時，Amazon EC2 Auto Scaling 會呼叫 Lifecycle hook，並嘗試立即啟動新 Spot 執行個體。

避免中斷替代 Spot 執行個體的風險提高

如果您使用 lowest-price 配置策略，替代 Spot 執行個體的中斷風險可能會提高。這是因為我們會在當時具有可用容量的最低價集區中啟動執行個體，即使您的替代 Spot 執行個體在啟動後很快就會中斷也一樣。為了避免提高中斷風險，我們強烈建議您不要使用 lowest-price 配置策略。

相反，我們建議使用 `price-capacity-optimized` 配置策略。此策略會在 Spot 集區中啟動最不可能中斷且價格可能最低的 Spot 執行個體。因此，它們在不久後的將來不太可能會中斷。

只有在可用性相同或更好時，Amazon EC2 Auto Scaling 才會啟動新執行個體

容量重新平衡的其中一個目標是改善 Spot 執行個體的可用性。如果現有的 Spot 執行個體收到重新平衡建議，只有新執行個體提供與現有執行個體相同或更好的可用性時，Amazon EC2 Auto Scaling 才會啟動新執行個體。如果新執行個體的中斷風險比現有執行個體更大，則 Amazon EC2 Auto Scaling 將不會啟動新執行個體。但是，Amazon EC2 Auto Scaling 將基於 Amazon EC2 Spot 服務提供的資訊，繼續評估 Spot 容量集區，並在可用性改善時啟動新執行個體。

如果 Amazon EC2 Auto Scaling 不主動啟動新執行個體，現有執行個體可能會中斷。發生這種情況時，Amazon EC2 Auto Scaling 會在收到 Spot 執行個體中斷通知後立即嘗試啟動新執行個體。無論新執行個體是否存在高中斷風險，都會進行此嘗試。

容量重新平衡不會增加 Spot 執行個體中斷率

當您啟用容量重新平衡時，不會增加 [Spot 執行個體中斷率](#) (當 Amazon EC2 需要恢復容量時回收的 Spot 執行個體數量)。但是，如果容量重新平衡偵測到執行個體有中斷風險，Amazon EC2 Auto Scaling 將立即嘗試啟動新的執行個體。因此，與在有風險的執行個體中斷後等待 Amazon EC2 Auto Scaling 啟動新的執行個體相比，可能會取代更多執行個體。

雖然您可以在啟用容量重新平衡的情況下取代更多執行個體，但您可以從主動式而非被動式中受益。這可讓您有更多時間在執行個體中斷之前採取行動。透過 [Spot 執行個體中斷通知](#)，您通常最多只有兩分鐘時間來正常關閉您的執行個體。透過容量重新平衡事先啟動新的執行個體，可讓現有程序更有可能在有風險的執行個體上完成作業。您也可以啟動執行個體關閉程序，防止在有風險的執行個體上排定新的作業，並為新啟動的執行個體作好準備以接管該應用程式。透過容量重新平衡的主動式取代，您可以從正常連續性中獲益。

下列理論範例示範了使用容量重新平衡的風險和益處：

- 下午 2:00 – 收到執行個體 A 的重新平衡建議，Amazon EC2 Auto Scaling 立即嘗試啟動取代執行個體 B，讓您有時間開始關機程序。
- 下午 2:30 – 收到執行個體 B 的重新平衡建議，會取代為執行個體 C，讓您有時間開始關機程序。
- 下午 2:32 – 如果未啟用容量重新平衡，並且如果在下午 2:32 仍未收到執行個體 A 的 Spot 執行個體中斷通知，則您最多只有兩分鐘時間採取動作。不過，執行個體 A 會一直持續執行，直到此時為止。

啟用容量重新平衡以主動取代具有風險的 Spot 執行個體

您可以使用 AWS 管理主控台 或 AWS CLI 來啟用 Auto Scaling 群組的容量重新平衡。啟用容量重新平衡時，Amazon EC2 Auto Scaling 會嘗試主動取代群組中收到 EC2 執行個體重新平衡建議的 Spot 執行個體。

啟用容量重新平衡 (主控台)

您可以在建立或更新 Auto Scaling 群組時啟用或停用容量重新平衡。

啟用新 Auto Scaling 群組的容量重新平衡

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。
3. 在步驟 1：選擇啟動範本或組態中，輸入 Auto Scaling 群組的名稱，選擇啟動範本，然後選擇下一步以繼續下一個步驟。
4. 在步驟 2：選擇執行個體啟動選項中，針對執行個體類型需求，選擇建立混合執行個體群組的設定。這包括可啟動的執行個體類型、執行個體購買選項，以及 Spot 和隨需執行個體的配置策略。依預設，這些設定並未設定。要進行設定，您必須選擇 Override launch template (覆寫啟動範本)。如需有關建立混合執行個體群組的詳細資訊，請參閱 [具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。
5. 在網路下，視需要選擇選項。驗證您想要使用的子網路是否位於不同的可用區域中。
6. 在配置策略區段下，選擇 Spot 的配置策略。勾選或清除容量重新平衡核取方塊，即可啟用或停用容量重新平衡。您唯有在執行個體購買選項區段中請求了要作為 Spot 執行個體啟動的 Auto Scaling 群組的百分比時，才能看到這個選項。
7. 建立 Auto Scaling 群組。
8. (選用) 視需要新增生命週期關聯。如需詳細資訊，請參閱[將生命週期掛鉤新增至 Auto Scaling 群組](#)。

啟用或停用現有 Auto Scaling 群組的容量重新平衡

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。頁面底部會開啟一個分割窗格。
3. 在 Details (詳細資訊) 索引標籤上，依次選擇 Allocation strategies (分配策略)、Edit (編輯)。

4. 在配置策略區段下，選取或清除容量重新平衡下的核取方塊，以啟用或停用容量重新平衡。
5. 選擇更新。

啟用容量重新平衡 (AWS CLI)

下列範例示範如何使用 AWS CLI 來啟用和停用容量重新平衡。

搭配以下參數使用 [create-auto-scaling-group](#) 或 [update-auto-scaling-group](#) 命令：

- `--capacity-rebalance` / `--no-capacity-rebalance` – 指出是否啟用容量重新平衡的布林值。

在呼叫 [create-auto-scaling-group](#) 命令前，您需要設定供 Auto Scaling 群組使用的啟動範本名稱。如需詳細資訊，請參閱 [為 Auto Scaling 群組建立啟動範本](#)。

Note

以下程序說明如何使用 JSON 或 YAML 格式組態檔案。如果您使用 AWS CLI 版本 1，則必須指定 JSON 格式的組態檔案。如果您使用 AWS CLI 版本 2，您可以指定 YAML 或 JSON 格式的組態檔案。

JSON

要建立和設定新的 Auto Scaling 群組

- 使用下列 [create-auto-scaling-group](#) 命令建立新的 Auto Scaling 群組並啟用容量重新平衡。此命令參考 JSON 檔案作為 Auto Scaling 群組的唯一參數。

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

如果您尚未擁有指定[混合執行個體政策](#)的 CLI 組態檔案，請建立一個。

在組態檔案中，將以下命令列新增至最上層 JSON 物件。

```
{  
  "CapacityRebalance": true  
}
```

以下是範例 config.json 檔案。

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredCapacity": 12,
  "MinSize": 12,
  "MaxSize": 15,
  "CapacityRebalance": true,
  "MixedInstancesPolicy": {
    "InstancesDistribution": {
      "OnDemandBaseCapacity": 0,
      "OnDemandPercentageAboveBaseCapacity": 25,
      "SpotAllocationStrategy": "price-capacity-optimized"
    },
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Default"
      },
      "Overrides": [
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        },
        {
          "InstanceType": "m5.large"
        },
        {
          "InstanceType": "m5a.large"
        },
        {
          "InstanceType": "c4.large"
        },
        {
          "InstanceType": "m4.large"
        },
        {
          "InstanceType": "c3.large"
        },
        {
          "InstanceType": "m3.large"
        }
      ]
    }
  }
}
```

```

        }
    ]
}
},
"TargetGroupARNs": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff",
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```

YAML

要建立和設定新的 Auto Scaling 群組

- 使用下列 [create-auto-scaling-group](#) 命令建立新的 Auto Scaling 群組並啟用容量重新平衡。此命令參考 YAML 檔案作為 Auto Scaling 群組的唯一參數。

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

將下命令列新增至 YAML 格式組態檔案。

```
CapacityRebalance: true
```

以下是範例 config.yaml 檔案。

```

---
AutoScalingGroupName: my-asg
DesiredCapacity: 12
MinSize: 12
MaxSize: 15
CapacityRebalance: true
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 0
    OnDemandPercentageAboveBaseCapacity: 25
    SpotAllocationStrategy: price-capacity-optimized
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large

```

- InstanceType: *c5a.large*
- InstanceType: *m5.large*
- InstanceType: *m5a.large*
- InstanceType: *c4.large*
- InstanceType: *m4.large*
- InstanceType: *c3.large*
- InstanceType: *m3.large*

TargetGroupARNs:

- *arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff*

VPCZoneIdentifier: *subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782*

要啟用現有 Auto Scaling 群組的容量重新平衡

- 使用下列 [update-auto-scaling-group](#) 命令來啟用容量重新平衡。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --capacity-rebalance
```

驗證是否已為 Auto Scaling 群組啟用容量重新平衡

- 使用下列 [describe-auto-scaling-groups](#) 命令來驗證是否已啟用容量重新平衡，並檢視詳細資訊。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下是回應範例。

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn",
      ...
      "CapacityRebalance": true
    }
  ]
}
```

停用容量重新平衡

使用 [update-auto-scaling-group](#) 命令與 `--no-capacity-rebalance` 選項來停用容量重新平衡。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--no-capacity-rebalance
```

相關資源

如需容量重新平衡的詳細資訊，請參閱 AWS 運算部落格上的[使用適用於 Amazon EC2 Auto Scaling 的新容量重新平衡功能主動管理 Spot 執行個體生命週期](#)。

如需 EC2 執行個體重新平衡建議的詳細資訊，請參閱《Amazon [EC2 使用者指南](#)》中的[EC2 執行個體重新平衡建議](#)。Amazon EC2

若要進一步了解生命週期關聯，請參閱下列資源。

- [教學課程：設定叫用 Lambda 函數的 lifecycle hook](#) (使用 EventBridge)
- [教學課程：使用資料指令碼和執行個體中繼資料來擷取生命週期狀態](#)

限制

- 只有在執行個體未受到縮減保護的情況下，Amazon EC2 Auto Scaling 才能取代接收重新平衡通知的執行個體。但是，縮減保護不會防止 Spot 中斷的終止。如需詳細資訊，請參閱[使用執行個體縮減保護來控制執行個體終止](#)。
- 容量重新平衡的支援可在所有 AWS 區域 提供 Amazon EC2 Auto Scaling 的商業區域使用，中東 (阿拉伯聯合大公國) 區域除外。

使用容量預留在特定可用區域中預留容量

Amazon EC2 隨需容量預留可讓您在特定可用區域中保留運算容量。若要開始將容量保留與 Auto Scaling 搭配使用，請先在特定可用區域中建立容量保留或容量保留群組。然後，您可以在建立容量保留偏好設定或更新現有群組時，將容量保留偏好設定新增至 Auto Scaling 群組。

若要建立容量保留，請參閱《Amazon EC2 使用者指南》中的[建立容量保留](#)。若要建立容量保留群組，請參閱《Amazon EC2 使用者指南》中的[建立容量保留群組](#)。

容量保留偏好設定

「容量保留」偏好設定可協助您在使用隨需容量之前，將使用容量保留中的預留容量設為優先，如此有助於您有效率地使用容量保留功能。您可以從下列容量保留偏好設定選項中選取：

- 預設 – Auto Scaling 會使用啟動範本或開放容量保留中的容量保留偏好設定。
- 無 – Auto Scaling 不會在容量保留中啟動執行個體。執行個體將以隨需容量執行。
- 僅限容量保留 – Auto Scaling 只會在容量保留或容量保留群組中啟動執行個體。如果容量無法使用，執行個體將無法啟動。
- 容量保留優先 – Auto Scaling 會在容量保留或容量保留群組中啟動執行個體。如果容量不可用，執行個體將以隨需容量執行。

如果您選取「僅限容量保留」或「容量保留優先」，則可以指定容量保留目標。

Note

您必須選取「容量保留」偏好設定。容量保留目標是選擇性的。

容量保留偏好設定和啟動範本的考量事項

如果選取「僅限容量保留」或「容量保留優先」，請將下列事項列入考量：

- 如果選取「僅限容量保留」或「容量保留優先」，則 Auto Scaling 會使用在 Auto Scaling 群組中指定的，而不是啟動範本中的容量保留目標。
- 如果選取「僅限容量保留」或「容量保留優先」，但未指定容量保留目標，則 Auto Scaling 會使用啟動範本容量保留目標，或開放的容量保留目標。

容量保留目標規格

如果您先選取容量保留或容量保留，可使用下列容量保留目標選項：

- 開啟 – Auto Scaling 會在任何開啟的容量保留中啟動執行個體。如果您只選取容量保留，但無法使用容量，則執行個體將無法啟動。如果您先選取容量保留，但無法使用容量，執行個體將以隨需容量啟動。
- 指定容量保留 – Auto Scaling 會在指定的容量保留中啟動執行個體。如果您只選取容量保留，但無法使用容量，則執行個體將無法啟動。如果您先選取容量保留，但無法使用容量，執行個體將以隨需容量啟動。
- 指定容量保留資源群組 – Auto Scaling 會在指定容量保留資源群組中的開放容量保留中啟動執行個體。如果您只選取容量保留，但無法使用容量，則執行個體將無法啟動。如果您先選取容量保留，但無法使用容量，執行個體將以隨需容量啟動。

可用區域餘額和容量保留

即使您使用容量保留偏好設定，Auto Scaling 仍會優先考慮可用區域餘額。這表示：

- 使用 `capacity-reservations-first` – Auto Scaling 群組會先在 AZs 之間平均分配執行個體，然後在可用時使用容量預留。如果可用區域無法使用容量預留，執行個體將以隨需方式啟動，以維持可用區域平衡。
- 使用 `capacity-reservations-only` – Auto Scaling 群組只會使用容量預留。根據可用容量預留的位置，這可能會導致 AZs 之間的執行個體分佈不均勻。

範例

如果您在 AZ-a 中有 10 個容量預留、AZ-b 中有 3 個、AZ-c 中有 1 個，且所需的容量為 9 個執行個體：

- 將 `capacity-reservations-first` 用於 9 個執行個體將導致每個 AZ 有 3 個執行個體（維護 AZ 平衡），其中有些執行個體可能以隨需執行。
- 將 `capacity-reservations-only` 用於 9 個執行個體會導致根據可用容量預留的分佈不均勻。

將容量保留偏好設定與 Auto Scaling 群組搭配使用

若要搭配 Auto Scaling 群組使用容量保留，您必須先建立容量保留或容量保留資源群組。然後，您可以在建立容量保留偏好設定或更新現有群組時，將容量保留偏好設定新增至 Auto Scaling 群組。

若要建立容量保留，請參閱《Amazon EC2 使用者指南》中的[建立容量保留](#)。若要建立容量保留群組，請參閱《Amazon EC2 使用者指南》中的[建立容量保留群組](#)。

當您建立或編輯 Auto Scaling 群組時，請使用下列其中一種方法來使用容量保留偏好設定。

Console

在新群組上使用容量保留偏好設定（主控台）

1. 遵循 中的指示[使用 Amazon EC2 啟動精靈建立 Auto Scaling 群組](#)並完成程序中的每個步驟，直到步驟 3。
2. 在設定群組大小和擴展頁面的其他容量設定、容量保留偏好設定下，選取容量保留偏好設定。如需容量保留偏好設定的詳細資訊，請參閱 [容量保留偏好設定](#)。
3. 繼續[使用 Amazon EC2 啟動精靈建立 Auto Scaling 群組](#)中的步驟。

AWS CLI

在新群組上使用容量保留偏好設定 (AWS CLI)

將 `--capacity-reservation-specification` 參數新增至 [create-auto-scaling-group](#) 命令。

1. 指定容量保留偏好設定。如需詳細資訊，請參閱[容量保留偏好設定](#)。
2. 指定容量保留目標。如果選取「僅限容量保留」或「容量保留優先」，但未指定容量保留目標，則 Auto Scaling 會使用啟動範本容量保留目標，或開放的容量保留目標。

Console

在現有群組上使用容量保留偏好設定（主控台）

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在螢幕上方的導覽列中，選擇您在建立 Auto Scaling 群組時所在的 AWS 區域。
3. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

4. 在詳細資訊索引標籤的容量保留偏好設定下，選擇編輯。
5. 在額外容量設定、容量保留偏好設定下，選取容量保留偏好設定。如需容量保留偏好設定的詳細資訊，請參閱 [容量保留偏好設定](#)。
6. 選擇更新。

AWS CLI

在現有群組上使用容量保留偏好設定 (AWS CLI)

將 `--capacity-reservation-specification` 參數新增至 [update-auto-scaling-group](#) 命令。

1. 指定容量保留偏好設定。如需詳細資訊，請參閱[容量保留偏好設定](#)。
2. 指定容量保留目標。如果選取「僅限容量保留」或「容量保留優先」，但未指定容量保留目標，則 Auto Scaling 會使用啟動範本容量保留目標，或開放的容量保留目標。

使用 EC2 Auto Scaling 進行可中斷的容量保留

可中斷容量預留代表 AWS 組織內隨需容量預留擁有者提供給您的備用容量。這些保留適用於可中斷的工作負載，因為容量擁有者可以隨時透過 2 分鐘的中斷通知回收容量，之後執行個體就會終止。如需可中斷容量保留及其運作方式的概觀，請參閱《Amazon EC2 使用者指南》中的[可中斷容量保留](#)。

若要開始使用，您可以從帳戶中的任何隨需容量保留配置未使用的容量，以建立可中斷的保留。接著，您可以建立啟動範本，指定可中斷的容量保留 ID 和適當的市場類型。最後，建立 Auto Scaling 群組以使用您建立的啟動範本。

Note

- 可中斷容量保留不支援混合執行個體群組。
- 可中斷的容量保留會顯示在您的帳戶中，並將新屬性 `interruptible` 設為 `true`。

主題

- [操作準則](#)
- [在啟動範本中指定可中斷的容量保留](#)
- [限制](#)
- [相關資源](#)

操作準則

以下是在 Auto Scaling 群組中使用可中斷容量保留時應遵循的基本操作準則。

- 從可中斷容量保留啟動的執行個體，可在回收容量時以 2 分鐘通知終止。設計您的應用程式來處理正常關機。
- Amazon EC2 會在終止執行個體前 2 分鐘傳送 EventBridge 通知。雖然 Auto Scaling 不會自動回應這些事件，但您可以設定自訂自動化來回應中斷通知。
- 當執行個體因容量回收而終止時，Auto Scaling 的運作狀態檢查程式會偵測終止，並根據您群組的組態自動啟動替換執行個體。
- 指向特定版本的啟動範本，而不是 `$Default` 或 `$Latest` 版本，以確保一致性。

Note

當回收可中斷的容量保留執行個體時，Auto Scaling 群組的擴展活動將會顯示：taken out of service in response to an Amazon EC2 health check that indicated it had been terminated or stopped。Auto Scaling 會嘗試以與未通過運作狀態檢查的任何執行個體相同的方式取代執行個體。

在啟動範本中指定可中斷的容量保留

若要建立以 Auto Scaling 群組特定可中斷容量保留為目標的啟動範本，請使用下列其中一種方法：

Console

在啟動範本中指定可中斷的容量保留（主控台）

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在頂端導覽列上，選取您有可中斷容量保留 AWS 區域的。
3. 在導覽窗格的 Instances (執行個體) 底下，選擇 Launch Templates (啟動範本)。
4. 選擇建立啟動範本，然後建立啟動範本。視需要包含 AMI 的 ID、執行個體類型和任何其他啟動範本設定。
5. 展開進階詳細資訊區段以檢視進階設定。
6. 針對購買選項，選擇可中斷的容量保留。
7. 針對容量保留，選擇依 ID 的目標，然後針對容量保留 - 依 ID 的目標，選擇現有可中斷容量保留的容量保留 ID。
8. 當您完成時，請選擇建立啟動範本。

如需使用啟動範本建立 Auto Scaling 群組的說明，請參閱 [使用啟動範本建立 Auto Scaling 群組](#)。

AWS CLI

在啟動範本中指定可中斷的容量保留 (AWS CLI)

使用下列 [create-launch-template](#) 命令來建立啟動範本，以指定現有的可中斷容量保留 ID。將每個使用者輸入預留位置替換為自己的資訊。

```
aws ec2 create-launch-template --launch-template-name my-template-for-interruptible-  
cr \
```

```
--version-description AutoScalingVersion1 --region us-east-2 \  
--launch-template-data file://config.json
```

 Tip

如果此命令擲回錯誤，請確定您已在 AWS CLI 本機將 更新為最新版本。

config.json 的內容：

```
{  
  "ImageId": "ami-04d5cc9b88example",  
  "InstanceType": "t3.nano",  
  "SecurityGroupIds": [  
    "sg-903004f88example"  
  ],  
  "KeyName": "MyKeyPair",  
  "InstanceMarketOptions": {  
    "MarketType": "interruptible-capacity-reservation"  
  },  
  "CapacityReservationSpecification": {  
    "CapacityReservationTarget": {  
      "CapacityReservationId": "cr-02168da1478b509e0"  
    }  
  }  
}
```

下列為範例輸出：

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

您可以使用下列 [describe-launch-template-versions](#) 命令來驗證與啟動範本相關聯的中斷容量保留 ID：

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

Auto Scaling 群組層級組態

您可以選擇性地在 Auto Scaling 群組層級指定容量保留設定，以覆寫啟動範本設定：

- 容量保留偏好設定：僅capacity-reservations-only支援可中斷的容量保留。不支援capacity-reservations-first偏好設定。
- 目標規格：您可以在群組層級指定不同的可中斷容量保留 ID，以覆寫啟動範本設定。

限制

- 可中斷容量保留不支援混合執行個體政策。
- 您必須在啟動範本中指定正確的市場類型 (interruptible-capacity-reservation) 以及容量保留 ID。

相關資源

- 如需有關建立和管理可中斷容量保留的資訊，請參閱《Amazon EC2 使用者指南》中的[可中斷容量保留](#)。
- 如需搭配 Auto Scaling 使用容量保留的一般資訊，請參閱 [使用容量預留在特定可用區域中預留容量](#)。
- 如需 Auto Scaling 群組運作狀態檢查的資訊，請參閱 [Auto Scaling 群組中執行個體的運作狀態檢查](#)。

使用 從命令列建立 Auto Scaling 群組 AWS CloudShell

在[支援的 AWS 區域](#)中，您可以 AWS CloudShell 針對直接從 啟動的瀏覽器型預先驗證 Shell，使用 執行 AWS CLI 命令 AWS 管理主控台。您可以使用您偏好的 shell (Bash、PowerShell 或 Z shell) 對 服務執行 AWS CLI 命令。

您可以使用下列兩種方法 AWS 管理主控台 之一，AWS CloudShell 從 啟動：

- 選擇主控台導覽列上的 AWS CloudShell 圖示。該圖示位於搜尋方塊的右側。
- 使用主控台瀏覽列上的搜尋方塊來搜尋 CloudShell，然後選擇 CloudShell 選項。

第一次在新的瀏覽器視窗中 AWS CloudShell 啟動時，歡迎面板會顯示並列出主要功能。關閉此面板後，狀態更新會顯示，同時 Shell 會設定並轉寄您的主控台憑證。出現命令提示時，表示 Shell 已準備好開始互動。

如需此服務的詳細資訊，請參閱 [AWS CloudShell 使用者指南](#)。

使用 建立 Auto Scaling 群組 AWS CloudFormation

Amazon EC2 Auto Scaling 已與 整合 AWS CloudFormation，此服務可協助您模型化和設定 AWS 資源，以減少建立和管理資源和基礎設施的時間。您可以建立範本來描述您想要的所有 AWS 資源（例如 Auto Scaling 群組），並為您 CloudFormation 佈建和設定這些資源。

使用 時 CloudFormation，您可以重複使用範本來一致且重複地設定 Amazon EC2 Auto Scaling 資源。描述您的資源一次，然後在多個 AWS 帳戶 和 區域中逐一佈建相同的資源。

Amazon EC2 Auto Scaling 和 CloudFormation 範本

若要為 Amazon EC2 Auto Scaling 及相關服務佈建和設定資源，您必須了解 [CloudFormation 範本](#)。範本是以 JSON 或 YAML 格式化的文本檔案。這些範本說明您要在 CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 CloudFormation 設計工具來協助您開始使用 CloudFormation 範本。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [什麼是 CloudFormation 設計工具？](#)。

若要開始建立您自己的 Amazon EC2 Auto Scaling 堆疊範本，請完成下列任務：

- 使用 [AWS::EC2::LaunchTemplate](#) 建立啟動範本。
- 使用 [AWS::AutoScaling::AutoScalingGroup](#) 建立 Auto Scaling 群組。

如需示範如何在 Application Load Balancer 後方部署 Auto Scaling 群組的演練，請參閱《AWS CloudFormation 使用者指南》中的 [演練：建立可擴展且達到負載平衡的應用程式](#)。

您可以在 AWS CloudFormation 使用者指南的下列章節中找到建立 Auto Scaling 群組和相關資源的範本程式碼片段的其他實用範例：

- [Amazon EC2 Auto Scaling 資源類型參考](#)
- [使用 設定 Amazon EC2 Auto Scaling 資源 AWS CloudFormation](#)

進一步了解 CloudFormation

若要進一步了解 CloudFormation，請參閱下列資源：

- [AWS CloudFormation](#)
- [AWS CloudFormation 使用者指南](#)
- [CloudFormation API 參考](#)
- [AWS CloudFormation 命令列界面使用者指南](#)

使用 取得執行個體類型建議 AWS Compute Optimizer

AWS 提供 Amazon EC2 執行個體類型建議，透過使用 支援的 功能，協助您改善效能、節省成本或兩者 AWS Compute Optimizer。您可以使用這些建議來決定是否要移至 Auto Scaling 群組中的新執行個體類型。

若要提出建議，Compute Optimizer 會分析您現有的執行個體規格和最近的指標歷史記錄。然後，編譯的資料會用來建議哪些 Amazon EC2 執行個體類型進行最佳化，以處理現有的效能工作負載。建議會連同每小時執行個體定價一起傳回。

Note

若要從 Compute Optimizer 取得建議，您必須先選擇加入 Compute Optimizer。如需詳細資訊，請參閱《AWS Compute Optimizer使用者指南》中的 [AWS Compute Optimizer 入門](#)。

目錄

- [限制](#)
- [調查結果](#)
- [檢視建議](#)
- [評估建議的考量](#)

限制

Compute Optimizer 會針對 Auto Scaling 群組中設定為啟動和執行 M、C、R、T 及 X 執行個體類型的執行個體產生建議。不過，它不會對採用 AWS Graviton2 處理器的 -g 執行個體類型（例如 C6g）和具有較高網路頻寬效能的 -n 執行個體類型（例如 M5n）產生建議。

Auto Scaling 群組也必須設定為執行單一執行個體類型（即沒有混合執行個體類型），不得連接至擴展政策，並且對所需容量、最小容量和最大容量（即具有固定數量執行個體的 Auto Scaling 群組）具有相同的值。Compute Optimizer 會針對 Auto Scaling 群組中符合所有這些組態需求的執行個體產生建議。

調查結果

Compute Optimizer 會將其 Auto Scaling 群組問題清單分類如下：

- Not optimized (未最佳化)：當 Compute Optimizer 找出可為您的工作負載提供更好效能的建議時，Auto Scaling 群組會被視為未最佳化。
- Optimized (已最佳化)：當 Compute Optimizer 根據選擇的執行個體類型判定群組已正確佈建為執行您的工作負載時，Auto Scaling 群組會被視為已最佳化。對於已最佳化的資源，Compute Optimizer 有時可能會建議新一代的執行個體類型。
- None (無)：沒有適合此 Auto Scaling 群組的建議。如果選擇加入 Compute Optimizer 的時間少於 12 小時，或 Auto Scaling 群組執行時間少於 30 小時，或 Compute Optimizer 不支援 Auto Scaling 群組或執行個體類型時，可能會發生這種情況。如需詳細資訊，請參閱 [限制](#) 一節。

檢視建議

在選擇加入 Compute Optimizer 之後，您可以檢視它針對您的 Auto Scaling 群組所產生的問題清單和建議。如果您最近才選擇加入，則最多 12 小時都不會提供任何建議。

檢視針對 Auto Scaling 群組所產生的建議

1. 在 <https://console.aws.amazon.com/compute-optimizer/> 開啟運算最佳化工具主控台。

儀表板頁面即會開啟。

2. 選擇 View recommendations for all Auto Scaling groups (檢視所有 Auto Scaling 群組的建議)。
3. 選取 Auto Scaling 群組。
4. 選擇 View detail (檢視詳細資訊)。

檢視會根據預設表格設定，變更為在預先設定的檢視中最多顯示三個不同的執行個體建議。它也會為 Auto Scaling 群組提供最近的 CloudWatch 指標資料 (CPU 平均使用率、平均網路輸入和平均網路輸出)。

決定您是否要使用其中一項建議。決定是否要最佳化以改善效能、降低成本，或是結合這兩者。

若要變更 Auto Scaling 群組中的執行個體類型，請更新啟動範本或 Auto Scaling 群組以使用新的啟動組態。現有執行個體會繼續使用先前的組態。若要更新現有執行個體，請終止它們，以便由 Auto Scaling 群組取代；或允許自動擴展，以便根據您的[終止政策](#)用新的執行個體逐漸取代舊的執行個體。

Note

使用執行個體生命週期上限和執行個體重新整理功能，您也可以取代 Auto Scaling 群組中現有的執行個體，從而啟動使用啟動範本或啟動組態的新執行個體。如需詳細資訊，請參閱 [根據執行個體生命週期上限來取代 Auto Scaling 執行個體](#) 和 [使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。

評估建議的考量

移至新的執行個體類型之前，請考慮下列事項：

- 建議不會預測您的使用量。建議是根據您最近 14 天期間的歷史使用情況而定。請務必選擇符合未來用量需求的執行個體類型。
- 著重於圖形化指標，以判斷實際使用量是否低於執行個體容量。您也可以檢視 CloudWatch 中的指標資料 (平均、峰值、百分比)，以進一步評估您的 EC2 執行個體建議。例如，注意 CPU 百分比指標在白天的變化，以及是否有需要應付的尖峰時期。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[檢視可用指標](#)。
- Compute Optimizer 可能會提供高載效能執行個體的建議，亦即 T3、T3a 和 T2 執行個體。如果您定期突增超過基準，請確定您可以根據新執行個體類型的 vCPU 繼續執行此操作。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[高載效能執行個體的 CPU 點數和基準效能](#)。
- 如果您購買的是預留執行個體，隨需執行個體可能會以預留執行個體方式計費。變更目前的執行個體類型之前，請先評估對預留執行個體使用率和涵蓋範圍的影響。
- 盡可能考慮轉換為較新一代的執行個體。

- 移轉至不同的執行個體系列時，請確定目前的執行個體類型和新的執行個體類型是相容的，例如在虛擬化、架構或網路類型方面。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[調整執行個體大小的相容性](#)。
- 最後，請考慮針對每個建議所提供的效能風險評分。效能風險指出您可能需要花費的精力，以驗證建議的執行個體類型是否符合工作負載的效能需求。我們也建議在進行任何變更之前和之後進行嚴格的負載和效能測試。

其他資源

除了本頁面上的主題之外，請參閱下列資源：

- [Amazon EC2 執行個體類型](#)
- [AWS Compute Optimizer 使用者指南](#)

使用 Elastic Load Balancing 在 Auto Scaling 群組中分配傳入應用程式流量

Elastic Load Balancing 會自動在您執行的所有 EC2 執行個體中分配傳入的應用程式流量。Elastic Load Balancing 以最佳化路由傳送流量來協助管理傳入請求，讓所有執行個體均不會負擔過重。若要搭配 Auto Scaling 群組使用 Elastic Load Balancing，請[將負載平衡器連接到 Auto Scaling 群組](#)。這樣會將群組註冊到負載平衡器，該負載平衡器會作為 Auto Scaling 群組所有 Web 流量的單一聯絡點。

在將 Elastic Load Balancing 與 Auto Scaling 群組搭配使用時，不需要將個別 EC2 執行個體註冊到負載平衡器。由 Auto Scaling 群組啟動的執行個體會自動註冊到負載平衡器。同樣地，由 Auto Scaling 群組終止的執行個體也會自動從負載平衡器取消註冊。

在將負載平衡器連接到 Auto Scaling 群組後，您可以設定 Auto Scaling 群組使用 Elastic Load Balancing 指標 (例如每個目標的 Application Load Balancer 請求計數)，以便隨著需求波動而擴展群組中的執行個體數目。

您也可以將 Elastic Load Balancing 運作狀態檢查新增到 Auto Scaling 群組，以便 Amazon EC2 Auto Scaling 根據這些額外的運作狀態檢查識別和取代運作狀態不佳的執行個體。此外，您還可以建立 CloudWatch 警示，在目標群組的狀態良好的主機數目低於允許值時通知您。

目錄

- [Elastic Load Balancing 類型](#)
- [準備連接 Elastic Load Balancing 負載平衡器](#)

- [將 Elastic Load Balancing 負載平衡器連接至 Auto Scaling 群組](#)
- [從主控台設定 Application Load Balancer 或 Network Load Balancer](#)
- [確認負載平衡器的連接狀態](#)
- [新增可用區域](#)
- [移除可用區域](#)
- [從 Auto Scaling 群組分離目標群組或 Classic Load Balancer](#)
- [使用 使用 Elastic Load Balancing 的範例 AWS CLI](#)

Elastic Load Balancing 類型

Elastic Load Balancing 提供四種可與 Auto Scaling 群組一起使用的負載平衡器類型：Application Load Balancer、Network Load Balancer、Gateway Load Balancer 和 Classic Load Balancer。

負載平衡器類型的設定方式有關鍵的差異。若採用 Application Load Balancer、Network Load Balancer 和 Gateway Load Balancer，執行個體會作為目標註冊到目標群組，並將流量路由到目標群組。若採用 Classic Load Balancer，執行個體會直接註冊到負載平衡器。

Application Load Balancer

在應用程式層 (HTTP/HTTPS) 的路由及負載平衡並支援以路徑為基礎的路由。Application Load Balancer 可將請求路由至 Virtual Private Cloud (VPC) 中一或多個已註冊目標 (例如 EC2 執行個體) 上的連接埠。

Network Load Balancer

在傳輸層 (TCP/UDP 第 4 層) 的路由和負載平衡，其以擷取自第 4 層標頭的地址資訊為基礎。Network Load Balancer 可以處理流量暴增、保留用戶端的來源 IP，並在負載平衡器生命週期中使用固定 IP。

Gateway Load Balancer

將流量分配到一組設備執行個體。為第三方虛擬設備 (例如防火牆、入侵偵測和預防系統以及其他設備) 提供擴展性、可用性和簡易性。Gateway Load Balancer 可與支援 GENEVE 通訊協定的虛擬設備搭配使用。由於需要進行額外的技術整合，請務必在選擇 Gateway Load Balancer 之前閱讀使用者指南。

Classic Load Balancer

在傳輸層 (TCP/SSL) 或應用程式層 (HTTP/HTTPS) 進行路由及負載平衡。

若要深入了解可用的不同類型的負載平衡器，請參閱下列資源：

- [什麼是 Elastic Load Balancing ?](#)
- [什麼是 Application Load Balancer ?](#)
- [什麼是 Network Load Balancer ?](#)
- [什麼是 Gateway Load Balancer ?](#)
- [什麼是 Classic Load Balancer ?](#)

準備連接 Elastic Load Balancing 負載平衡器

將 Elastic Load Balancing 負載平衡器連接至 Auto Scaling 群組之前，您必須先完成下列先決條件：

- 您必須已建立用於將流量路由至 Auto Scaling 群組的負載平衡器和目標群組。

有兩種方法可以建立負載平衡器和目標群組：

- 使用 Elastic Load Balancing – 在建立 Auto Scaling 群組之前，請遵循 Elastic Load Balancing 文件中的程序來建立和設定負載平衡器和目標群組。略過註冊 Amazon EC2 執行個體的步驟。當您將目標群組連接至 Auto Scaling 群組時，Amazon EC2 Auto Scaling 會自動負責註冊（和取消註冊）執行個體。如需詳細資訊，請參閱《Elastic Load Balancing 使用者指南》中的 [Elastic Load Balancing 入門](#)。
- 使用 Amazon EC2 Auto Scaling – 從 Amazon EC2 Auto Scaling 主控台建立、設定和連接具有基本組態的負載平衡器和目標群組。如需詳細資訊，請參閱[從主控台設定 Application Load Balancer 或 Network Load Balancer](#)。
- 在建立負載平衡器之前，請了解您需要的負載平衡器類型。如需詳細資訊，請參閱[Elastic Load Balancing 類型](#)。
- 負載平衡器及其目標群組必須與 Auto Scaling 群組位於相同的 AWS 帳戶 VPC 和區域。
- 目標群組必須指定 instance 的目標類型。在使用 Auto Scaling 群組時，您無法指定 ip 的目標類型。
- 如果 Auto Scaling 群組的啟動範本不包含正確的安全群組，以允許來自負載平衡器的必要傳入流量，您必須更新啟動範本。建議採用的規則取決於負載平衡器的類型和負載平衡器使用的後端類型。例如，若要將流量路由傳送至 Web 伺服器，則允許從負載平衡器的連接埠 80 上傳入 HTTP 存取。修改啟動範本時，現有執行個體不會以新設定更新。若要更新現有的執行個體，您可以啟動執行個體重新整理來取代執行個體。如需詳細資訊，請參閱[使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。

- 啟動範本中的安全群組也必須允許從 Elastic Load Balancing 正確連接埠上的負載平衡器存取，才能執行其運作狀態檢查。
- 在 Gateway Load Balancer 後方部署虛擬設備時，啟動範本中的 Amazon Machine Image (AMI) 必須指定支援 GENEVE 通訊協定的 AMI ID，以允許 Auto Scaling 群組與 Gateway Load Balancer 交換流量。此外，啟動範本中的安全群組必須允許連接埠 6081 上的 UDP 流量。

Tip

如果您的 Bootstrap 指令碼需要一段時間才能完成，可以選擇將啟動 lifecycle hook 新增至 Auto Scaling 群組，以在 Bootstrap 指令碼成功完成且執行個體上的應用程式已準備好接受流量之前，先延遲在負載平衡器後再註冊執行個體。初始在 Amazon EC2 Auto Scaling 主控台中建立 Auto Scaling 群組時，無法新增 lifecycle hook。不過，您可以在建立群組之後新增生命週期關聯。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling lifecycle hook](#)。

設定目標的運作狀態檢查

您可以為向 Elastic Load Balancing 負載平衡器註冊的目標設定運作狀態檢查，以確保它們能夠正確處理流量。特定步驟會根據您使用的負載平衡器類型而有所不同。如需詳細資訊，請參閱下列資源：

- Application Load Balancer – 請參閱 Application Load Balancer 使用者指南中的[目標群組的運作狀態檢查](#)。
- Network Load Balancer – 請參閱 Network Load Balancer 使用者指南中的[目標群組的運作狀態檢查](#)。
- Gateway Load Balancer – 請參閱《Gateway Load Balancer 使用者指南》中的[目標群組的運作狀態檢查](#)。
- Classic Load Balancer – 請參閱[Classic Load Balancer 使用者指南中的設定 Classic Load Balancer 的運作狀態檢查](#)。

根據預設，Amazon EC2 Auto Scaling 不會將執行個體視為運作狀態不佳，如果執行個體未通過 Elastic Load Balancing 運作狀態檢查，則會予以取代。Auto Scaling 群組的預設運作狀態檢查只進行 EC2 運作狀態檢查。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

若要讓 Amazon EC2 Auto Scaling 取代 Elastic Load Balancing 回報運作狀態不佳的執行個體，您可以將 Auto Scaling 群組設定為使用 Elastic Load Balancing 運作狀態檢查。如此一來，如果執行個體未通過 Amazon EC2 Auto Scaling 就會將執行個體視為運作狀態不佳。Elastic Load Balancing

如果將多個負載平衡器目標群組或 Classic Load Balancer 連接到群組，上述所有項目都必須回報執行個體狀態良好，群組才會將該執行個體視為狀態良好。假如上述任一個項目回報執行個體運作狀態不佳，Auto Scaling 群組便會替換該執行個體，即使其他回報其運作狀態良好也一樣。

如需如何為 Auto Scaling 群組啟用這些運作狀態檢查的資訊，請參閱 [將 Elastic Load Balancing 負載平衡器連接至 Auto Scaling 群組](#)。

Note

為了確保這些運作狀態檢查盡快開始，請確定您群組的運作狀態檢查寬限期未設定太高，但足夠讓 Elastic Load Balancing 運作狀態檢查判斷目標是否可用於處理請求。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。

將 Elastic Load Balancing 負載平衡器連接至 Auto Scaling 群組

本主題說明如何將 Elastic Load Balancing 負載平衡器連接至 Auto Scaling 群組。其中也說明如何開啟 Elastic Load Balancing 運作狀態檢查，讓 Amazon EC2 Auto Scaling 取代 Elastic Load Balancing 回報運作狀態不佳的執行個體。

預設情況下，Amazon EC2 Auto Scaling 只會根據 Amazon EC2 運作狀態檢查來取代運作狀態不佳或無法連接的執行個體。如果您開啟 Elastic Load Balancing 運作狀態檢查，則如果您連接到 Auto Scaling 群組的任何 Elastic Load Balancing 負載平衡器回報運作狀態不佳，Amazon EC2 Auto Scaling 可以取代執行中的執行個體。

如需將 Application Load Balancer 連接至 Auto Scaling 群組的教學課程，請參閱 [教學課程：設定擴展和負載平衡應用程式](#)。

Important

繼續操作之前，請先完成上一節中的所有[先決條件](#)。

目錄

- [連接目標群組或 Classic Load Balancer](#)
- [分離目標群組或 Classic Load Balancer](#)

連接目標群組或 Classic Load Balancer

當您建立或更新 Auto Scaling 群組時，您可以連接一或多個目標群組或 Classic Load Balancer。當您連接 Application Load Balancer、Network Load Balancer 或 Gateway Load Balancer 時，您會連接目標群組，而不是負載平衡器本身。

請依照本節中的步驟，使用主控台執行下列操作：

- 將目標群組或 Classic Load Balancer 連接至 Auto Scaling 群組
- 開啟 Elastic Load Balancing 的運作狀態檢查

若要在建立新 Auto Scaling 群組時連接現有負載平衡器

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在畫面頂端的導覽列上，選擇您在其中建立負載平衡器的 AWS 區域。
3. 選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。
4. 在步驟 1 和步驟 2 中，選擇所需選項，然後繼續進行步驟 3：設定進階選項。
5. 針對 Load balancing (負載平衡)，選擇 Attach to an existing load balancer (連接到現有負載平衡器)。
6. 在 Attach to an existing load balancer (連接到現有負載平衡器) 下，執行下列其中一項操作：
 - a. 對於 Application Load Balancer、Network Load Balancer 和 Gateway Load Balancer：

選擇 Choose from your load balancer target groups (從負載平衡器目標群組中選擇)，然後從 Existing load balancer target groups (現有負載平衡器目標群組) 欄位中選擇目標群組。
 - b. 對於 Classic Load Balancer：

選擇 Choose from Classic Load Balancers (從 Classic Load Balancer 中選擇)，然後從 Classic Load Balancer 欄位中選擇您的負載平衡器。
7. (選用) 在運作狀態檢查、其他運作狀態檢查類型上，選取開啟 Elastic Load Balancing 運作狀態檢查。
8. (選用) 針對運作狀態檢查寬限期，輸入時長，以秒為單位。這是 Amazon EC2 Auto Scaling 在進入 InService 狀態後檢查執行個體的運作狀態之前需要等待的時間。如需詳細資訊，請參閱 [設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。
9. 繼續建立 Auto Scaling 群組。建立 Auto Scaling 群組之後，您的執行個體會自動註冊到負載平衡器。

將現有負載平衡器連接到建立後的 Auto Scaling 群組

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在整合索引標籤上，選擇負載平衡、編輯。
4. 在 Load balancing (負載平衡) 下方，執行下列其中一項操作：
 - a. 針對 Application, Network or Gateway Load Balancer target groups (Application、Network 或 Gateway Load Balancer 目標群組)，選取其核取方塊並選擇目標群組。
 - b. 針對 Classic Load Balancer，選取其核取方塊並選擇您的負載平衡器。
5. 選擇更新。

當您完成連接負載平衡器時，您可以選擇開啟使用負載平衡器的運作狀態檢查。

開啟 Elastic Load Balancing 運作狀態檢查

1. 在 Details (詳細資訊) 索引標籤上，選擇 Health checks (運作狀態檢查)、Edit (編輯)。
2. 在運作狀態檢查、其他運作狀態檢查類型上，選取開啟 Elastic Load Balancing 運作狀態檢查。
3. 對於運作狀態檢查寬限期，輸入時間量，以秒為單位。這是 Amazon EC2 Auto Scaling 在進入 InService 狀態後檢查執行個體的運作狀態之前需要等待的時間。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。
4. 選擇更新。

Note

您可以在使用 AWS CLI 連接負載平衡器時，監控負載平衡器的狀態。當 Amazon EC2 Auto Scaling 已成功註冊執行個體，且至少有一個已註冊的執行個體通過了運作狀態檢查時，您就會獲得 InService 狀態。如需詳細資訊，請參閱[確認負載平衡器的連接狀態](#)。

分離目標群組或 Classic Load Balancer

如果不再需要負載平衡器，請按照下列步驟從 Auto Scaling 群組分開它。

將負載平衡器從群組中分開

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Load balancing (負載平衡)、Edit (編輯)。
4. 在 Load balancing (負載平衡) 下方，執行下列其中一項操作：
 - a. 針對 Application, Network or Gateway Load Balancer target groups (Application、Network 或 Gateway Load Balancer 目標群組)，選擇目標群組旁的刪除圖示 (X)。
 - b. 針對 Classic Load Balancer，選擇負載平衡器旁的刪除圖示 (X)。
5. 選擇更新。

當您完成分離目標群組時，您可以關閉 Elastic Load Balancing 運作狀態檢查。

關閉 Elastic Load Balancing 運作狀態檢查

1. 在 Details (詳細資訊) 索引標籤上，選擇 Health checks (運作狀態檢查)、Edit (編輯)。
2. 對於運作狀態檢查、其他運作狀態檢查類型，取消選取開啟 Elastic Load Balancing 運作狀態檢查。
3. 選擇更新。

從主控台設定 Application Load Balancer 或 Network Load Balancer

在建立 Auto Scaling 群組時，請使用下列程序來建立和連接 Application Load Balancer 或 Network Load Balancer。

若要在建立新的 Auto Scaling 群組時建立和連接新的負載平衡器

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。
3. 在步驟 1 和步驟 2 中，選擇所需選項，然後繼續進行步驟 3：設定進階選項。
4. 針對 Load balancing (負載平衡)，選擇 Attach to a new load balancer (連接到新的負載平衡器)。

- a. 針對 Attach to a new load balancer (連接到新的負載平衡器) 下的 Load balancer type (負載平衡器類型)，選擇建立 Application Load Balancer 或 Network Load Balancer。
 - b. 針對 Load balancer name (負載平衡器名稱)，輸入負載平衡器的名稱，或保留預設名稱。
 - c. 針對 Load balancer scheme (負載平衡器機制)，選擇建立面向公有網際網路的負載平衡器，或保留內部負載平衡器的預設值。
 - d. 針對 Availability Zones and subnets (可用區域與子網路)，選取您在其中啟用 EC2 執行個體的各個可用區域的公有子網路。(這些資訊會在步驟 2 中預先填入。)
 - e. 針對 Listeners and routing (接聽程式和路由)，請更新接聽程式的連接埠號碼 (若有必要)，再在 Default routing (預設路由) 下選擇 Create a target group (建立目標群組)。您也可以從下拉式清單中選擇現有目標群組。
 - f. 如果在最後一個步驟中選擇 Create a target group (建立目標群組)，對於 New target group name (新目標群組名稱)，請輸入目標群組的名稱，或保留預設名稱。
 - g. 若要將標籤新增至負載平衡器，選取 Add tag (新增標籤)，並提供各標的標籤鍵和標籤值。
5. (選用) 在運作狀態檢查、其他運作狀態檢查類型上，選取開啟 Elastic Load Balancing 運作狀態檢查。
 6. (選用) 針對運作狀態檢查寬限期，輸入時長，以秒為單位。這是 Amazon EC2 Auto Scaling 在進入 InService 狀態後檢查執行個體的運作狀態之前需要等待的時間。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。
 7. 繼續建立 Auto Scaling 群組。建立 Auto Scaling 群組之後，您的執行個體會自動註冊到負載平衡器。

Note

建立 Auto Scaling 群組之後，您可以使用 Elastic Load Balancing 主控台來建立其他接聽程式。如需建立使用安全通訊協定 (例如 HTTPS) 的接聽程式或 UDP 接聽程式，便會用到此功能。只要使用不同的連接埠，您就可以將更多接聽程式新增至現有負載平衡器。

確認負載平衡器的連接狀態

連接負載平衡器後，它會進入 Adding 狀態，同時註冊該群組中的執行個體。在群組內所有執行個體都註冊完成後，它會進入 Added 狀態。在至少有一個註冊的執行個體通過運作狀態檢查後，它就會進入 InService 狀態。當負載平衡器進入 InService 狀態後，Amazon EC2 Auto Scaling 可以終止和替換任何回報為運作狀態不佳的執行個體。如果沒有註冊的執行個體通過運作狀態檢查 (例如，因為運

作狀態檢查的設定錯誤)，負載平衡器不會進入 InService 狀態。Amazon EC2 Auto Scaling 不會終止和替換執行個體。

當您分離負載平衡器後，它會進入 Removing 狀態，同時取消註冊該群組的執行個體。取消註冊後的執行個體仍繼續執行作業。依預設，Application Load Balancer、Network Load Balancer 和 Gateway Load Balancer 會啟用連接耗盡 (取消註冊延遲)。如果啟用了連接耗盡，Elastic Load Balancing 會等待執行中的請求完成或最大逾時過期 (以兩者之中先發生者為準)，再取消註冊執行個體。

您可以使用 AWS Command Line Interface (AWS CLI) AWS SDKs 來驗證連接狀態。您無法從主控台確認連接狀態。

使用 AWS CLI 驗證連接狀態

以下 [describe-traffic-sources](#) 命令會傳回指定 Auto Scaling 群組之所有流量來源的連接狀態。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

此範例會傳回連接至 Auto Scaling 群組的 Elastic Load Balancing 目標群組的 ARN，以及 State 元素中目標群組的連接狀態。

```
{
  "TrafficSources": [
    {
      "Identifier": "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456",
      "State": "InService",
      "Type": "elbv2"
    }
  ]
}
```

新增可用區域

若要利用地理備援的安全性和可靠性，請讓 Auto Scaling 群組跨越您工作所在區域中的多個可用區域，並連接負載平衡器以跨越這些可用區域分配傳入流量。

當一個可用區域運作狀態不佳或無法使用時，Amazon EC2 Auto Scaling 會在受影響的可用區域中啟動新的執行個體。當運作狀態不佳的可用區域恢復到良好狀態時，Amazon EC2 Auto Scaling 會自動為 Auto Scaling 群組所有可用區域重新平均分配應用程式執行個體。為此，Amazon EC2 Auto Scaling

會嘗試在可用區域中使用最少的執行個體啟動新的執行個體。不過，若是嘗試失敗了，Amazon EC2 Auto Scaling 會試著在其他可用區域中啟動執行個體，直到成功為止。

Elastic Load Balancing 會為您為負載平衡器啟用的各個可用區域建立負載平衡器節點。若您啟用負載平衡器跨區域負載平衡功能，則每個負載平衡器節點會將流量平均分配至所有啟用的可用區域內已註冊的執行個體。若顯示跨區域負載平衡，每個負載平衡器節點會平均地將請求僅分配到其可用區域中已註冊的執行個體。

在建立 Auto Scaling 群組時，您必須指定至少一個可用區域。之後，您可以透過新增可用區域到 Auto Scaling 群組，並啟用負載平衡器的可用區域 (若負載平衡器提供支援) 來擴展您應用程式的可用性。

限制

若要為負載平衡器更新啟用的可用區域，您必須注意下列限制：

- 在為負載平衡器啟用可用區域時，您要從該可用區域指定一個子網路。請注意，您可以為負載平衡器在每個可用區域啟用最多一個子網路。
- 對於面向網際網路的負載平衡器，您為負載平衡器指定的子網路必須至少有八個可用的 IP 地址。
- 針對 Application Load Balancer，您必須啟用至少兩個可用區域。
- 針對 Network Load Balancer，您無法停用已啟用的可用區域，但可以啟用其他區域。
- 對於閘道負載平衡器，您無法停用已啟用的可用區域，但您可以啟用其他可用區域。

按照下列步驟將 Auto Scaling 群組和負載平衡器擴展至其他可用區域內的子網路。

如要新增可用區域

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Network (網路)、Edit (編輯)。
4. 在 Subnet (子網路) 中，選擇與要新增到 Auto Scaling 群組的可用區域對應的子網路。
5. 選擇更新。
6. 若要為您的負載平衡器更新可用區域，以使其與您的 Auto Scaling 群組共用相同的可用區域，請完成以下步驟：
 - a. 在導覽窗格的 Load Balancing (負載平衡器)，選擇 Load Balancer (負載平衡器)。

- b. 選擇您的負載平衡器。
- c. 執行以下任意一項：
 - 對於 Application Load Balancer 與 Network Load Balancer：
 1. 在 Description (描述) 索引標籤上，針對 Availability Zones (可用區域) 選擇 Edit (編輯)。
 2. 在 Edit subnets (編輯子網路) 頁面上，針對 Availability Zones (可用區域)，選取要新增的可用區域核取方塊。如果該區域只有一個子網路，則會選取該子網路。如果該區域有多個子網路，則請選取其中一個子網路。
 - 對於 VPC 中的 Classic Load Balancer：
 1. 在 Instances (執行個體) 索引標籤上，選擇 Edit Availability Zones (編輯可用區域)。
 2. 在 Add and Remove Subnets (新增和移除子網路) 頁面上，針對 Available subnets (可用的子網路)，使用新增 (+) 圖示選擇該子網路。子網路會移至 Selected subnets (選取子網路) 下。
- d. 選擇儲存。

相關資源

在您變更可用區域時，Amazon EC2 Auto Scaling 會重新平衡群組。這表示取代和重新分配某些執行個體。如需詳細資訊，請參閱[範例：跨可用區域分配執行個體](#)。

如果您在可用區域中註冊的目標尚未啟用負載平衡器，則負載平衡器不會將流量路由到這些目標。如需詳細資訊，請參閱 Elastic Load Balancing 使用者指南中的 [Elastic Load Balancing 的運作方式](#)。

移除可用區域

若要從 Auto Scaling 群組和負載平衡器移除可用區域，請按下列步驟操作。

若要移除可用區域

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Network (網路)、Edit (編輯)。

4. 在 Subnet (子網路) 中，針對要從 Auto Scaling 群組移除的可用區域對應的子網路，請選擇刪除 (X) 圖示。如果該區域有多個子網路，則選擇每個子網路的刪除 (X) 圖示。
5. 選擇更新。
6. 若要為您的負載平衡器更新可用區域，以使其與您的 Auto Scaling 群組共用相同的可用區域，請完成以下步驟：
 - a. 在導覽窗格的 Load Balancing (負載平衡器)，選擇 Load Balancer (負載平衡器)。
 - b. 選擇您的負載平衡器。
 - c. 執行以下任意一項：
 - 針對 Application Load Balancer：
 1. 在 Description (描述) 索引標籤上，針對 Availability Zones (可用區域) 選擇 Edit (編輯)。
 2. 在 Edit subnets (編輯子網路) 頁面上，對於 Availability Zones (可用區域)，取消勾選要移除之該可用區域子網路的核取方塊。
 - 對於 VPC 中的 Classic Load Balancer：
 1. 在 Instances (執行個體) 索引標籤上，選擇 Edit Availability Zones (編輯可用區域)。
 2. 在 Add and Remove Subnets (新增和移除子網路) 頁面上，對於 Available subnets (可用的子網路)，使用刪除 (-) 圖示以移除該子網路。子網路移至 Available subnets (可用的子網路) 下。
 - d. 選擇儲存。

從 Auto Scaling 群組分離目標群組或 Classic Load Balancer

如果不再需要負載平衡器，請按照下列步驟從 Auto Scaling 群組分開它。

將負載平衡器從群組中分開

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

Auto Scaling groups (Auto Scaling 群組) 頁面底部會開啟一個分割窗格。

3. 在 Details (詳細資訊) 索引標籤上，選擇 Load balancing (負載平衡)、Edit (編輯)。
4. 在 Load balancing (負載平衡) 下方，執行下列其中一項操作：

- a. 針對 Application, Network or Gateway Load Balancer target groups (Application、Network 或 Gateway Load Balancer 目標群組)，選擇目標群組旁的刪除圖示 (X)。
 - b. 針對 Classic Load Balancer，選擇負載平衡器旁的刪除圖示 (X)。
5. 選擇更新。

當您完成分離目標群組時，您可以關閉 Elastic Load Balancing 運作狀態檢查。

關閉 Elastic Load Balancing 運作狀態檢查

1. 在 Details (詳細資訊) 索引標籤上，選擇 Health checks (運作狀態檢查)、Edit (編輯)。
2. 對於運作狀態檢查、其他運作狀態檢查類型，取消選取開啟 Elastic Load Balancing 運作狀態檢查。
3. 選擇更新。

使用 使用 Elastic Load Balancing 的範例 AWS CLI

使用 AWS Command Line Interface (AWS CLI) 來連接、分離和描述負載平衡器和目標群組、新增和移除 Elastic Load Balancing 運作狀態檢查，以及變更已啟用的可用區域。

本主題顯示執行 Amazon EC2 Auto Scaling 常見任務的 AWS CLI 命令範例。

Important

如需其他命令範例，請參閱 AWS CLI 命令參考中的 [aws elbv2](#) 和 [aws elb](#)。

目錄

- [連接您的目標群組或 Classic Load Balancer](#)
- [描述您的目標群組或 Classic Load Balancer](#)
- [新增 Elastic Load Balancing 運作狀態檢查](#)
- [變更您的可用區域](#)
- [分離您的目標群組或 Classic Load Balancer](#)
- [移除 Elastic Load Balancing 運作狀態檢查](#)
- [舊版命令](#)

連接您的目標群組或 Classic Load Balancer

使用下列 [create-auto-scaling-group](#) 命令建立 Auto Scaling 群組，同時透過指定其 Amazon Resource Name (ARN) 來連接目標群組。目標群組可與 Application Load Balancer、Network Load Balancer 或 Gateway Load Balancer 關聯。

取代 `--auto-scaling-group-name`、`--vpc-zone-identifier`、`--min-size` 和 `--max-size` 的範例值。對於 `--launch-template` 選項，請使用 Auto Scaling 群組的啟動範本名稱和版本取代 `my-launch-template` 和 `1`。對於 `--traffic-sources` 選項，請用 Application Load Balancer、Network Load Balancer 或 Gateway Load Balancer 目標群組的 ARN 取代範例 ARN。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --min-size 1 --max-size 5 \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/12345678EXAMPLE1"
```

建立 Auto Scaling 群組後，使用 [attach-traffic-sources](#) 命令，將額外的目標群組連接至該 Auto Scaling 群組。

下列命令可將另一個目標群組新增至同一群組。

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/12345678EXAMPLE2"
```

或者，若要將 Classic Load Balancer 連接至您的群組，請在使用 `create-auto-scaling-group` 或 `attach-traffic-sources` 時指定 `--traffic-sources` 和 `--type` 選項，如下列範例中所示。將 `my-classic-load-balancer` 取代為 Classic Load Balancer 的名稱。對於 `--type` 選項，請指定值為 `elb`。

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

描述您的目標群組或 Classic Load Balancer

若要描述連結至 Auto Scaling 群組的負載平衡器或目標群組，請使用下列 [describe-traffic-sources](#) 命令。用您的群組名稱取代 `my-asg`。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

此範例會傳回您連接至 Auto Scaling 群組的 Elastic Load Balancing 目標群組的 ARN。

```
{
  "TrafficSources": [
    {
      "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE1",
      "State": "InService",
      "Type": "elbv2"
    },
    {
      "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE2",
      "State": "InService",
      "Type": "elbv2"
    }
  ]
}
```

如需 State 輸出中欄位的說明，請參閱 [確認負載平衡器的連接狀態](#)。

新增 Elastic Load Balancing 運作狀態檢查

若要將 Elastic Load Balancing 運作狀態檢查新增至 Auto Scaling 群組對執行個體執行的運作狀態檢查，請使用下列 [update-auto-scaling-group](#) 命令並指定 **ELB** 作為 `--health-check-type` 選項的值。用您的群組名稱取代 *my-asg*。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
--health-check-type "ELB"
```

新執行個體通常需要時間進行短暫暖機，才能通過運作狀態檢查。如果寬限期未提供足夠的暖機時間，則執行個體可能無法準備好提供流量。Amazon EC2 Auto Scaling 可能會認為這些執行個體運作狀態不佳並加以替換。

若要更新運作狀態檢查寬限期，請在使用 `update-auto-scaling-group` 時選用 `--health-check-grace-period` 選項，如下列範例中所示。如果發現新執行個體的運作狀態不佳，請將 **300** 取代為將新執行個體終止之前保持其服務狀態的秒數。

```
--health-check-grace-period 300
```

如需詳細資訊，請參閱 [Auto Scaling 群組中的執行個體運作狀態檢查](#)。

變更您的可用區域

變更您的可用區域時，您應留意一些限制。如需詳細資訊，請參閱[新增可用區域](#)。

變更 Application Load Balancer 或 Network Load Balancer 的可用區域

1. 在您變更負載平衡器的可用區域之前，最好先更新 Auto Scaling 群組的可用區域，以確認指定區域中的執行個體類型是否可用。

若要更新 Auto Scaling 群組的可用區域，請使用下列 [update-auto-scaling-group](#) 命令。使用要啟用的可用區域中的子網路 ID 取代範例子網路 ID。指定的子網路會取代先前啟用的子網路。用您的群組名稱取代 *my-asg*。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --vpc-zone-identifier "subnet-41767929,subnet-cb663da2,subnet-8360a9e7"
```

2. 請使用以下 [describe-auto-scaling-groups](#) 命令以驗證新子網路中的執行個體已啟動。如果執行個體已啟動，則您會看到執行個體及其狀態的清單。用您的群組名稱取代 *my-asg*。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. 使用下列 [set-subnets](#) 命令來指定負載平衡器的子網路。使用要啟用的可用區域中的子網路 ID 取代範例子網路 ID。每個可用區域只能指定一個子網路。指定的子網路會取代先前啟用的子網路。使用負載平衡器的 ARN 取代 *my-lb-arn*。

```
aws elbv2 set-subnets --load-balancer-arn my-lb-arn \
  --subnets subnet-41767929 subnet-cb663da2 subnet-8360a9e7
```

變更 Classic Load Balancer 的可用區域

1. 在您變更負載平衡器的可用區域之前，最好先更新 Auto Scaling 群組的可用區域，以確認指定區域中的執行個體類型是否可用。

若要更新 Auto Scaling 群組的可用區域，請使用下列 [update-auto-scaling-group](#) 命令。使用要啟用的可用區域中的子網路 ID 取代範例子網路 ID。指定的子網路會取代先前啟用的子網路。用您的群組名稱取代 *my-asg*。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --vpc-zone-identifier "subnet-41767929,subnet-cb663da2"
```


2. 請使用以下 [describe-auto-scaling-groups](#) 命令以驗證新子網路中的執行個體已啟動。如果執行個體已啟動，則您會看到執行個體及其狀態的清單。用您的群組名稱取代 *my-asg*。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. 使用以下 [attach-load-balancer-to-subnets](#) 命令來啟用 Classic Load Balancer 的新可用區域。使用要啟用的可用區域中的子網路 ID 取代範例子網路 ID。使用負載平衡器的名稱取代 *my-lb*。

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-lb \  
--subnets subnet-cb663da2
```

若要停用可用區域，請執行下列 [detach-load-balancer-from-subnets](#) 命令。使用要停用的可用區域中的子網路 ID 取代範例子網路 ID。使用負載平衡器的名稱取代 *my-lb*。

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-lb \  
--subnets subnet-8360a9e7
```

分離您的目標群組或 Classic Load Balancer

如果不再需要目標群組，請使用以下 [detach-load-balancer-target-groups](#) 命令從您的 Auto Scaling 群組分離該目標群組。

對於 `--auto-scaling-group-name` 選項，請用您的群組名稱取代 *my-asg*。對於 `--traffic-sources` 選項，請用 Application Load Balancer、Network Load Balancer 或 Gateway Load Balancer 目標群組的 ARN 取代範例 ARN。

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \  
--traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456"
```

若要從群組中分離 Classic Load Balancer，請指定 `--traffic-sources` 和 `--type` 選項，如下列範例所示。將 *my-classic-load-balancer* 取代為 Classic Load Balancer 的名稱。對於 `--type` 選項，請指定值為 **elb**。

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

移除 Elastic Load Balancing 運作狀態檢查

若要將 Elastic Load Balancing 運作狀態檢查從 Auto Scaling 群組中移除，請使用下列 [update-auto-scaling-group](#) 命令並指定 **EC2** 作為 `--health-check-type` 選項的值。用您的群組名稱取代 *my-asg*。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
--health-check-type "EC2"
```

如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

舊版命令

下列範例顯示如何使用舊版 CLI 命令連接、分離以及描述負載平衡器和目標群組。它們仍保留在本文件中，供任何想要使用的客戶做參考。我們會繼續支援舊版 CLI 命令，但建議您使用新的「流量來源」CLI 命令，該命令可以連接和分離多種流量來源類型。您可以在同一個 Auto Scaling 群組上同時使用舊版 CLI 命令和「流量來源」CLI 命令。

連接您的目標群組或 Classic Load Balancer (舊版)

連接目標群組

使用以下 [create-auto-scaling-group](#) 命令建立已連接目標群組的 Auto Scaling 群組。指定 Application Load Balancer、Network Load Balancer 或 Gateway Load Balancer 目標群組的 Amazon Resource Name (ARN)。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
--launch-template LaunchTemplateName=my-launch-template,Version='1' \
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
--target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456" \
--min-size 1 --max-size 5
```

下列 [attach-load-balancer-target-groups](#) 命令會將目標群組連接到現有 Auto Scaling 群組。

```
aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-name my-asg \
--target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456"
```

連接 Classic Load Balancer

使用以下 [create-auto-scaling-group](#) 命令建立已連接 Classic Load Balancer 的 Auto Scaling 群組。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
  --launch-configuration-name my-launch-config \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
  --load-balancer-names "my-load-balancer" \  
  --min-size 1 --max-size 5
```

以下 [attach-load-balancers](#) 命令會將指定的 Classic Load Balancer 連接到現有 Auto Scaling 群組。

```
aws autoscaling attach-load-balancers --auto-scaling-group-name my-asg \  
  --load-balancer-names my-lb
```

描述您的目標群組或 Classic Load Balancer (舊版)

描述目標群組

若要描述與 Auto Scaling 群組相關聯的目標群組，請使用 [describe-load-balancer-target-groups](#) 命令。下列範例列出 *my-asg* 的目標群組。

```
aws autoscaling describe-load-balancer-target-groups --auto-scaling-group-name my-asg
```

描述 Classic Load Balancer

若要描述與 Auto Scaling 群組相關聯的 Classic Load Balancer，請使用 [describe-load-balancers](#) 命令。下列範例會列出 *my-asg* 的 Classic Load Balancer。

```
aws autoscaling describe-load-balancers --auto-scaling-group-name my-asg
```

分離您的目標群組或 Classic Load Balancer (舊版)

分離目標群組

如果不再需要目標群組，請使用以下 [detach-load-balancer-target-groups](#) 命令從您的 Auto Scaling 群組分開該目標群組。

```
aws autoscaling detach-load-balancer-target-groups --auto-scaling-group-name my-asg \  
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456"
```

分離 Classic Load Balancer

如果不再需要 Classic Load Balancer，則可使用以下 [detach-load-balancers](#) 命令將其從您的 Auto Scaling 群組分開。

```
aws autoscaling detach-load-balancers --auto-scaling-group-name my-asg \  
--load-balancer-names my-lb
```

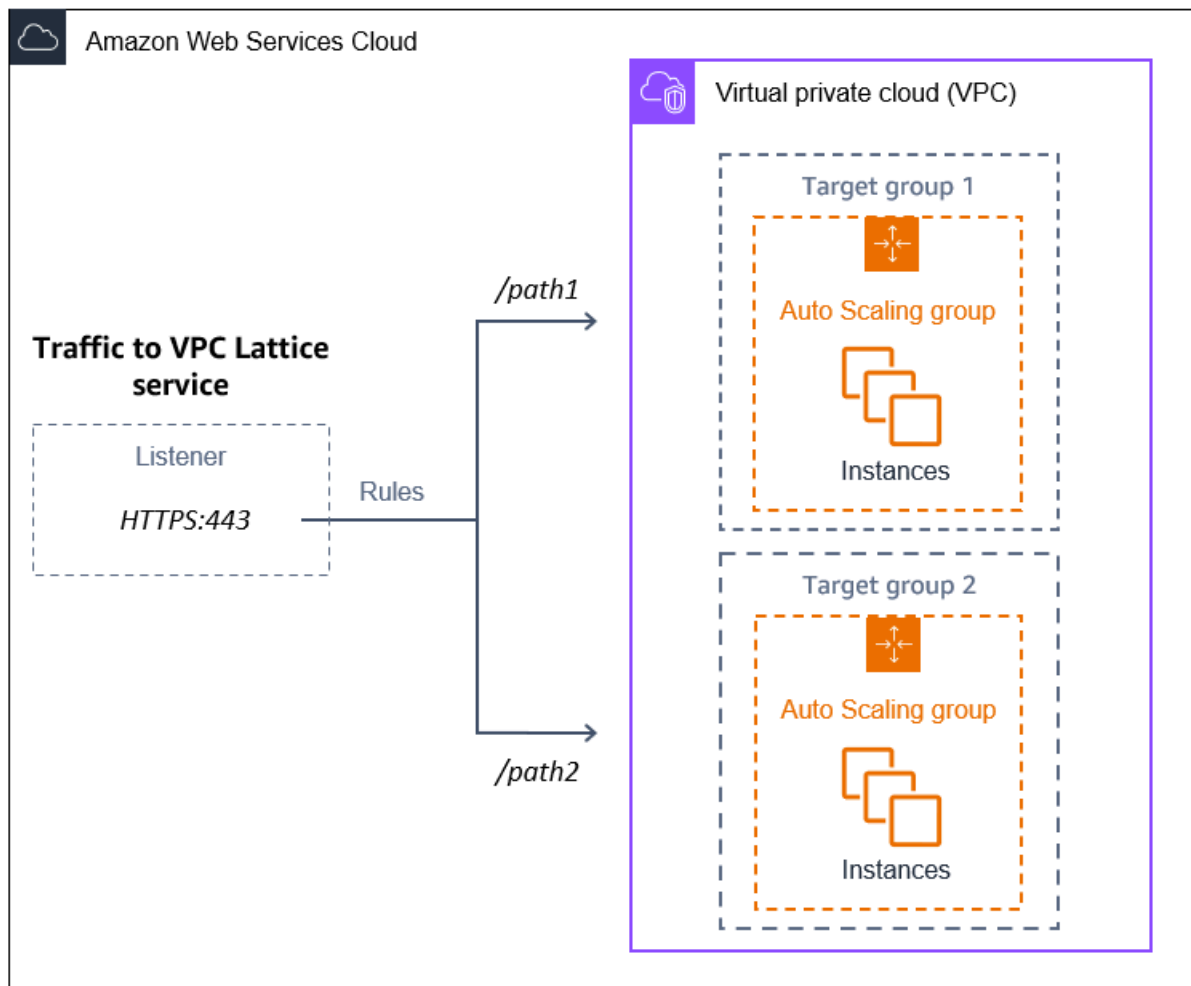
使用 VPC Lattice 目標群組管理流量流程

您可以使用 Amazon VPC Lattice 管理在不同資源 (例如 Auto Scaling 群組 或 Lambda 函數) 上執行的應用程式與服務之間的流量和 API 呼叫流程。VPC Lattice 是一項應用程式聯網服務，可讓您連接、保護和監控跨多個帳戶和虛擬私有雲端 (VPC) 的所有服務。若要進一步了解 VPC Lattice，請參閱《[什麼是 VPC Lattice？](#)》

若要開始使用 VPC Lattice，請先建立所需的 VPC Lattice 資源，使與服務網路相關聯的 VPC 資源能夠互相連結。這些資源包括服務、接聽程式、接聽程式規則和目標群組。

若要將 Auto Scaling 群組與 VPC Lattice 服務建立關聯，請為將請求路由至執行個體 ID 註冊的執行個體之服務建立目標群組，並且為服務加入接聽程式以傳送請求至目標群組。然後，將目標群組連接到 Auto Scaling 群組。Amazon EC2 Auto Scaling 會自動將 EC2 執行個體註冊為目標群組的目標。稍後，當 Amazon EC2 Auto Scaling 需要終止執行個體時，它會在終止前自動從目標群組取消註冊執行個體。

連接目標群組之後，這裡成為所有傳入請求連接至 Auto Scaling 群組的進入點。如下圖中的範例所示，接著便可以使用 VPC Lattice 服務指定的接聽程式規則，將傳入請求路由至適當的目標群組。



當流量透過 VPC Lattice 路由到 Auto Scaling 群組時，VPC Lattice 會使用循環配置負載平衡來平衡群組中執行個體之間的請求。VPC Lattice 還可監控其註冊執行個體的運作狀態，而且只會將流量路由到正常運作的執行個體。

若要確保您的執行個體可用於傳入的請求，您可以選擇將 VPC Lattice 運作狀態檢查新增至 Auto Scaling 群組。如此一來，若其中一個 EC2 執行個體發生失敗，Auto Scaling 群組會自動啟動新的執行個體將其取代。VPC Lattice 運作狀態檢查的行為類似於 Elastic Load Balancing 運作狀態檢查的行為。Auto Scaling 群組的預設運作狀態檢查只進行 EC2 運作狀態檢查。

若要進一步了解 VPC Lattice，請參閱 AWS 部落格上的[使用 Amazon VPC Lattice 簡化Service-to-Service連線、安全性和監控 – 現在已全面推出](#)。

目錄

- [準備將 VPC Lattice 目標群組連接到 Auto Scaling 群組](#)
- [將 VPC Lattice 目標群組連接到 Auto Scaling 群組](#)

- [確認 VPC Lattice 目標群組的连接狀態](#)

準備將 VPC Lattice 目標群組連接到 Auto Scaling 群組

將 VPC Lattice 目標群組連接到 Auto Scaling 群組之前，您必須完成以下先決條件：

- 您必須已經建立 VPC Lattice 服務網路、服務、接聽程式和目標群組。如需詳細資訊，請參閱《VPC Lattice 使用者指南》中的下列主題：
 - [服務網路](#)
 - [服務](#)
 - [接聽程式](#)
 - [目標群組](#)
- 目標群組必須與 Auto Scaling 群組位於相同的 AWS 帳戶 VPC 和區域。
- 目標群組必須指定 instance 的目標類型。在使用 Auto Scaling 群組時，您無法指定 ip 的目標類型。
- 您必須擁有足夠的 IAM 許可才能將目標群組連接到 Auto Scaling 群組。下列範例政策顯示連結和分離目標群組所需的最低必要許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:AttachTrafficSources",
        "autoscaling:DetachTrafficSources",
        "autoscaling:DescribeTrafficSources",
        "vpc-lattice:RegisterTargets",
        "vpc-lattice:DeregisterTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

- 如果 Auto Scaling 群組的啟動範本未包含 VPC Lattice 的正確設定 (例如相容的安全群組)，您必須更新啟動範本。修改啟動範本時，現有執行個體不會使用新的設定進行更新。若要更新現有的執行個

體，您可以啟動執行個體重新整理以取代執行個體。如需詳細資訊，請參閱[使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。

- 在您的 Auto Scaling 群組上啟用 VPC Lattice 運作狀態檢查之前，您可以設定應用程式型運作狀態檢查，以確認應用程式是否按預期回應。如需詳細資訊，請參閱《VPC Lattice 使用者指南》中的[目標群組的運作狀態檢查](#)一節。

安全群組：傳入和傳出規則

安全群組就像是防火牆，用於關聯的 EC2 執行個體，可在執行個體層級控制傳入及傳出流量。

Note

網路組態相當複雜，我們強烈建議您建立新的安全群組與 VPC Lattice 搭配使用。如果您需要聯絡他們支援，也可讓您更輕鬆地提供協助。以下幾節的內容是假設您遵循此建議而撰寫。若要深入了解如何為 VPC Lattice 建立可與 Auto Scaling 群組搭配使用的安全群組，請參閱《VPC Lattice 使用者指南》中的[使用安全群組控制流量](#)一節。若要對流量問題進行疑難排解，請參閱《VPC Lattice 使用者指南》以取得更多資訊。

如需有關如何建立安全群組的資訊，請參閱《Amazon EC2 使用者指南》中的[建立安全群組](#)，並使用下表來決定要選取的選項。

選項	Value	
名稱	您容易記住的名稱。	
Description	協助您識別安全群組的描述。	
VPC	與 Auto Scaling 群組相同的 VPC。	

傳入規則

當您建立安全群組時，它沒有傳入規則。將規則傳入安全群組之前，來自 VPC Lattice 服務網路內用戶端的流量都無法傳入您的執行個體。

若要允許 VPC Lattice 服務網路中的用戶端連結到 Auto Scaling 群組中的執行個體，您必須正確設定 Auto Scaling 群組的安全群組。在此情況下，請為其提供傳入規則，以允許來自 VPC Lattice AWS 受

管字首清單名稱的流量，而非特定 IP 地址。VPC Lattice 字首清單是 VPC Lattice 使用 CIDR 標記法表示的 IP 地址範圍。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[使用 AWS 受管字首清單](#)。

如需有關如何將規則新增至安全群組的資訊，請參閱《Amazon VPC 使用者指南》中的[設定安全群組規則](#)，並使用下表來決定要選取的選項。

選項	Value	
HTTP 規則	類型：HTTP 來源：com.am azonaws. <i>region</i> .vpc-lattice	
HTTPS 規則	Type (類型)：HTTPS 來源：com.am azonaws. <i>region</i> .vpc-lattice	

安全群組具有狀態：它允許從 VPC Lattice 服務網路內的用戶端流量傳送回 Auto Scaling 群組中的執行個體，然後將回應傳回先前離開的用戶端。

傳出規則

根據預設，安全群組會包含允許所有傳出流量的規則。您可以選擇移除此預設規則，並新增傳出規則以符合特定的安全需求。

限制

- 支援[混合執行個體群組](#)。如果您將 VPC Lattice 目標群組連接到具有混合執行個體政策的 Auto Scaling 群組，負載平衡演算法會將負載平均分配到所有可用資源，並假設執行個體的相似度足以處理相等的負載。

將 VPC Lattice 目標群組連接到 Auto Scaling 群組

本主題說明如何將 VPC Lattice 目標群組連接到 Auto Scaling 群組。同時也說明如何開啟 VPC Lattice 運作狀態檢查，讓 Amazon EC2 Auto Scaling 取代 VPC Lattice 報告為運作狀態不佳的執行個體。

預設情況下，Amazon EC2 Auto Scaling 只會根據 Amazon EC2 運作狀態檢查來取代運作狀態不佳或無法連接的執行個體。如果您開啟 VPC Lattice 運作狀態檢查，當您所連接之 Auto Scaling 群組有任何 VPC Lattice 目標群組報告為運作狀態不佳時，Amazon EC2 Auto Scaling 就可以取代執行中的執行個體。如需詳細資訊，請參閱[Auto Scaling 群組中的執行個體運作狀態檢查](#)。

Important

繼續操作之前，請先完成上一節中的所有[先決條件](#)。

連接 VPC Lattice 目標群組

建立或更新群組時，您可以將一或多個目標群組連接至 Auto Scaling 群組。

Console

請依照本節中的步驟，使用主控台執行下列操作：

- 將 VPC Lattice 目標群組連接到 Auto Scaling 群組
- 開啟 VPC Lattice 的運作狀態檢查

將 VPC Lattice 目標群組連接到新的 Auto Scaling 群組

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 在螢幕上方的導覽列中，選擇您建立目標群組所在的 AWS 區域。
3. 選擇 Create Auto Scaling group (建立 Auto Scaling 群組)。
4. 在步驟 1 和步驟 2 中，選擇所需選項，然後繼續進行步驟 3：設定進階選項。
5. 針對 VPC Lattice 整合選項，選擇連接至 VPC Lattice 服務。
6. 在選擇 VPC Lattice 目標群組下，選擇您的目標群組。
7. (選用) 針對運作狀態檢查、其他運作狀態檢查類型，選取開啟 VPC Lattice 運作狀態檢查。
8. (選用) 針對運作狀態檢查寬限期，輸入時長，以秒為單位。此時間量是 Amazon EC2 Auto Scaling 在進入 InService 狀態後檢查執行個體的運作狀態所需的等待時間。如需詳細資訊，請參閱[設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。
9. 繼續建立 Auto Scaling 群組。建立 Auto Scaling 群組之後，您的執行個體會自動註冊到 VPC Lattice 目標群組。

將 VPC Lattice 目標群組連接到現有的 Auto Scaling 群組

請按照下列步驟將服務的目標群組連接到現有 Auto Scaling 群組。

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取 Auto Scaling 群組旁的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在詳細資訊索引標籤上，選擇 VPC Lattice 整合選項、編輯。
4. 在 VPC Lattice 整合選項下，選擇連接至 VPC Lattice 服務。
5. 在選擇 VPC Lattice 目標群組下，選擇您的目標群組。
6. 選擇更新。

連接目標群組後，您可以選擇開啟並使用運作狀態檢查。

開啟 VPC Lattice 運作狀態檢查

1. 在 Details (詳細資訊) 索引標籤上，選擇 Health checks (運作狀態檢查)、Edit (編輯)。
2. 針對運作狀態檢查、其他運作狀態檢查類型，選取開啟 VPC Lattice 運作狀態檢查。
3. 針對運作狀態檢查寬限期，輸入時間量，以秒為單位。此時間量是 Amazon EC2 Auto Scaling 在進入 InService 狀態後檢查執行個體的運作狀態所需的等待時間。如需詳細資訊，請參閱 [設定 Auto Scaling 群組的運作狀態檢查寬限期](#)。
4. 選擇更新。

AWS CLI

請依照本節中的步驟使用 AWS CLI 來：

- 將 VPC Lattice 目標群組連接到 Auto Scaling 群組
- 開啟 VPC Lattice 的運作狀態檢查

將 VPC Lattice 目標群組連接到 Auto Scaling 群組

使用下列 [create-auto-scaling-group](#) 命令建立 Auto Scaling 群組，同時透過指定其 Amazon Resource Name (ARN) 來連接 VPC Lattice 目標群組。

取代 `--auto-scaling-group-name`、`--vpc-zone-identifier`、`--min-size` 和 `--max-size` 的範例值。在 `--launch-template` 選項中，使用您為註冊到 VPC Lattice 目標群組的執行個體所建立的啟動範本名稱和版本來取代 `my-launch-template` 和 `1`。在 `--traffic-sources` 選項中，使用 VPC Lattice 目標群組的 ARN 來取代範例 ARN。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --min-size 1 --max-size 5 \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/tg-0e2f2665eEXAMPLE"
```

建立 Auto Scaling 群組後，使用以下 [attach-traffic-sources](#) 命令，將 VPC Lattice 目標群組連接至該 Auto Scaling 群組。

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/tg-0e2f2665eEXAMPLE"
```

開啟 VPC Lattice 運作狀態檢查

如果您已為 VPC Lattice 目標群組設定應用程式型運作狀態檢查，則可以開啟這些運作狀態檢查。使用包含 `--health-check-type` 選項和 `VPC_LATTICE` 的值的 [create-auto-scaling-group](#) 或 [update-auto-scaling-group](#) 命令。若要指定 Auto Scaling 群組執行運作狀態檢查寬限期，請加入 `--health-check-grace-period` 選項並提供以秒為單位的值。

```
--health-check-type "VPC_LATTICE" --health-check-grace-period 60
```

分離 VPC Lattice 目標群組

如果不再需要使用 VPC Lattice，請按照下列步驟將目標群組從 Auto Scaling 群組分離。

Console

請依照本節中的步驟，使用主控台執行下列操作：

- 將 VPC Lattice 目標群組從 Auto Scaling 群組分離
- 關閉 VPC Lattice 運作狀態檢查

將 VPC Lattice 目標群組從 Auto Scaling 群組分離

1. 前往網址 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
2. 選取現有群組旁邊的核取方塊。

頁面底部會開啟一個分割窗格。

3. 在詳細資訊索引標籤上，選擇 VPC Lattice 整合選項、編輯。
4. 在 VPC Lattice 整合選項下，選擇目標群組旁的刪除 (X) 圖示。
5. 選擇更新。

分離目標群組後，您便可以關閉 VPC Lattice 運作狀態檢查。

關閉 VPC Lattice 運作狀態檢查

1. 在 Details (詳細資訊) 索引標籤上，選擇 Health checks (運作狀態檢查)、Edit (編輯)。
2. 針對運作狀態檢查、其他運作狀態檢查類型，取消選取開啟 VPC Lattice 運作狀態檢查。
3. 選擇更新。

AWS CLI

請依照本節中的步驟使用 AWS CLI 來：

- 將 VPC Lattice 目標群組從 Auto Scaling 群組分離
- 關閉 VPC Lattice 運作狀態檢查

如果不再需要目標群組，請使用 [detach-traffic-sources](#) 命令將目標群組從 Auto Scaling 群組分離。

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \  
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/tg-0e2f2665eEXAMPLE"
```

若要更新 Auto Scaling 群組的運作狀態檢查，使其不再使用 VPC Lattice 運作狀態檢查，請使用 [update-auto-scaling-group](#) 命令。包含 `--health-check-type` 選項和 **EC2** 的值。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
  --health-check-type "EC2"
```

確認 VPC Lattice 目標群組的连接狀態

將 VPC Lattice 目標群組連接至 Auto Scaling 群組後，該群組將進入 Adding 狀態並在群組內註冊執行個體。群組內所有執行個體都完成註冊後，它會進入 Added 狀態。在至少有一個註冊的執行個體通過運作狀態檢查後，它就會進入 InService 狀態。當目標群組進入 InService 狀態後，Amazon EC2 Auto Scaling 可以終止和取代任何報告為運作狀態不佳的執行個體。如果沒有註冊的執行個體通過運作狀態檢查 (例如，因為運作狀態檢查的設定錯誤)，目標群組就不會進入 InService 狀態。Amazon EC2 Auto Scaling 不會終止和替換執行個體。

當您分離服務的目標群組後，它會進入 Removing 狀態，同時取消在群組中註冊執行個體。取消註冊後的執行個體仍繼續執行作業。連接耗盡 (取消註冊延遲) 預設為啟用。如果啟用了連接耗盡，則 VPC Lattice 會等待傳輸中請求完成或最大逾時過期 (以先發生者為準)，再取消註冊執行個體。

您可以使用 AWS Command Line Interface (AWS CLI) AWS SDKs 來驗證連接狀態。您無法從主控台確認連接狀態。

使用 AWS CLI 驗證連接狀態

以下 [describe-traffic-sources](#) 命令會傳回指定 Auto Scaling 群組之所有流量來源的连接狀態。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

此範例會傳回連接至 Auto Scaling 群組的 VPC Lattice 目標群組的 ARN，以及 State 元素中目標群組的连接狀態。

```
{
  "TrafficSources": [
    {
      "Identifier": "arn:aws:vpc-lattice:region:account-id:targetgroup/tg-0e2f2665eEXAMPLE",
      "State": "InService",
      "Type": "vpc-lattice"
    }
  ]
}
```

使用 EventBridge 處理 Auto Scaling 事件

先前稱為 CloudWatch Events 的 Amazon EventBridge 可協助您設定事件驅動規則，以監控資源並啟動使用其他服務的目標動作 AWS。

Amazon EC2 Auto Scaling 的事件會以接近即時的方式傳送到 EventBridge。您可建立 EventBridge 規則來叫用程式設計動作和通知，以便回應這些事件。例如，當執行個體正在啟動或終止時，您可以叫用 AWS Lambda 函數來執行預先設定的任務。

EventBridge 規則的目標可以包含 AWS Lambda 函數、Amazon SNS 主題、API 目的地、其他事件匯流排 AWS 帳戶等。如需有關支援目標的資訊，請參閱《Amazon EventBridge 使用者指南》中的 [Amazon EventBridge 目標](#)。

首先請以使用 Amazon SNS 主題和 EventBridge 規則的範例來建立 EventBridge 規則。然後，當使用者啟動執行個體重新整理時，只要到達檢查點，Amazon SNS 就會透過電子郵件通知您。如需詳細資訊，請參閱[針對執行個體重新整理事件建立 EventBridge 規則](#)。

目錄

- [Amazon EC2 Auto Scaling 事件參考](#)
- [執行個體重新整理範例事件和模式](#)
- [暖集區範例事件和模式](#)
- [使用 Amazon EventBridge 規則來自動化動作](#)

Amazon EC2 Auto Scaling 事件參考

使用 Amazon EventBridge，您可以建立符合傳入事件的規則，並將其路由到目標以進行處理。

目錄

- [生命週期動作事件](#)
- [成功的擴展事件](#)
- [不成功的擴展事件](#)

生命週期動作事件

在將生命週期關聯新增至 Auto Scaling 群組時，Amazon EC2 Auto Scaling 會在執行個體轉換到等待狀態時，將事件傳送至 EventBridge。盡可能產生事件。

Event types (事件類型)

- [橫向擴展生命週期動作](#)
- [縮減生命週期動作](#)

橫向擴展生命週期動作

下列範例事件說明了由於啟動 lifecycle hook，Amazon EC2 Auto Scaling 已將執行個體移至 Pending:Wait 狀態。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

縮減生命週期動作

下列範例事件說明了由於終止 lifecycle hook，Amazon EC2 Auto Scaling 已將執行個體移至 Terminating:Wait 狀態。

Important

當 Auto Scaling 群組在縮減時將執行個體傳回暖集區時，執行個體傳回暖集區也可能會產生 EC2 Instance-terminate Lifecycle Action 事件。在縮減時將執行個體移至等待狀態時，交付的事件會以 WarmPool 作為 Destination 的值。如需詳細資訊，請參閱[Instance reuse policy](#)。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata": "additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

成功的擴展事件

下列範例顯示成功的擴展事件的事件類型。盡可能產生事件。

Event types (事件類型)

- [成功的橫向擴展事件](#)
- [成功的縮減事件](#)

成功的橫向擴展事件

下列範例事件說明 Amazon EC2 Auto Scaling 成功啟動執行個體。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
```

```

"time": "yyyy-mm-ddThh:mm:ssZ",
"region": "us-west-2",
"resources": [
  "auto-scaling-group-arn",
  "instance-arn"
],
"detail": {
  "StatusCode": "InProgress",
  "Description": "Launching a new EC2 instance: i-12345678",
  "AutoScalingGroupName": "my-asg",
  "ActivityId": "87654321-4321-4321-4321-210987654321",
  "Details": {
    "Availability Zone": "us-west-2b",
    "Subnet ID": "subnet-12345678"
  },
  "RequestId": "12345678-1234-1234-1234-123456789012",
  "StatusMessage": "",
  "EndTime": "yyyy-mm-ddThh:mm:ssZ",
  "EC2InstanceId": "i-1234567890abcdef0",
  "StartTime": "yyyy-mm-ddThh:mm:ssZ",
  "Cause": "description-text",
  "Origin": "EC2",
  "Destination": "AutoScalingGroup"
}
}

```

成功的縮減事件

下列範例事件說明 Amazon EC2 Auto Scaling 成功終止執行個體。

Important

當 Auto Scaling 群組在縮減時將執行個體傳回暖集區時，將執行個體傳回暖集區也可能會產生 EC2 Instance Terminate Successful 事件。執行個體成功傳回暖集區時，交付的事件會以 WarmPool 作為 Destination 的值。如需詳細資訊，請參閱[Instance reuse policy](#)。

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Successful",
  "source": "aws.autoscaling",

```

```

"account": "123456789012",
"time": "yyyy-mm-ddThh:mm:ssZ",
"region": "us-west-2",
"resources": [
  "auto-scaling-group-arn",
  "instance-arn"
],
"detail": {
  "StatusCode": "InProgress",
  "Description": "Terminating EC2 instance: i-12345678",
  "AutoScalingGroupName": "my-asg",
  "ActivityId": "87654321-4321-4321-4321-210987654321",
  "Details": {
    "Availability Zone": "us-west-2b",
    "Subnet ID": "subnet-12345678"
  },
  "RequestId": "12345678-1234-1234-1234-123456789012",
  "StatusMessage": "",
  "EndTime": "yyyy-mm-ddThh:mm:ssZ",
  "EC2InstanceId": "i-1234567890abcdef0",
  "StartTime": "yyyy-mm-ddThh:mm:ssZ",
  "Cause": "description-text",
  "Origin": "AutoScalingGroup",
  "Destination": "EC2"
}
}

```

不成功的擴展事件

下列範例顯示不成功的擴展事件的事件類型。盡可能產生事件。

Event types (事件類型)

- [不成功的橫向擴展事件](#)
- [不成功的縮減事件](#)

不成功的橫向擴展事件

下列範例事件說明 Amazon EC2 Auto Scaling 無法啟動執行個體。

```

{
  "version": "0",

```

```

    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Instance Launch Unsuccessful",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-west-2",
    "resources": [
      "auto-scaling-group-arn",
      "instance-arn"
    ],
    "detail": {
      "StatusCode": "Failed",
      "AutoScalingGroupName": "my-asg",
      "ActivityId": "87654321-4321-4321-4321-210987654321",
      "Details": {
        "Availability Zone": "us-west-2b",
        "Subnet ID": "subnet-12345678"
      },
      "RequestId": "12345678-1234-1234-1234-123456789012",
      "StatusMessage": "message-text",
      "EndTime": "yyyy-mm-ddThh:mm:ssZ",
      "EC2InstanceId": "i-1234567890abcdef0",
      "StartTime": "yyyy-mm-ddThh:mm:ssZ",
      "Cause": "description-text",
      "Origin": "EC2",
      "Destination": "AutoScalingGroup"
    }
  }
}

```

不成功的縮減事件

下列範例事件說明 Amazon EC2 Auto Scaling 無法終止執行個體。

Important

當 Auto Scaling 群組在縮減時將執行個體傳回暖集區時，無法將執行個體傳回暖集區也可能會產生 EC2 Instance Terminate Unsuccessful 事件。執行個體無法傳回暖集區時，交付的事件會以 WarmPool 作為 Destination 的值。如需詳細資訊，請參閱[Instance reuse policy](#)。

```
{
```

```

"version": "0",
"id": "12345678-1234-1234-1234-123456789012",
"detail-type": "EC2 Instance Terminate Unsuccessful",
"source": "aws.autoscaling",
"account": "123456789012",
"time": "yyyy-mm-ddThh:mm:ssZ",
"region": "us-west-2",
"resources": [
  "auto-scaling-group-arn",
  "instance-arn"
],
"detail": {
  "StatusCode": "Failed",
  "AutoScalingGroupName": "my-asg",
  "ActivityId": "87654321-4321-4321-4321-210987654321",
  "Details": {
    "Availability Zone": "us-west-2b",
    "Subnet ID": "subnet-12345678"
  },
  "RequestId": "12345678-1234-1234-1234-123456789012",
  "StatusMessage": "message-text",
  "EndTime": "yyyy-mm-ddThh:mm:ssZ",
  "EC2InstanceId": "i-1234567890abcdef0",
  "StartTime": "yyyy-mm-ddThh:mm:ssZ",
  "Cause": "description-text",
  "Origin": "AutoScalingGroup",
  "Destination": "EC2"
}
}

```

執行個體重新整理範例事件和模式

使用 Amazon EventBridge，您可以建立符合傳入事件的規則，並將其路由到目標以進行處理。

目錄

- [執行個體重新整理事件](#)
- [執行個體重新整理取代根磁碟區生命週期事件](#)

執行個體重新整理事件

下列範例顯示執行個體重新整理功能的事件。盡可能產生事件。

Event types (事件類型)

- [達到檢查點](#)
- [已啟動執行個體重新整理](#)
- [執行個體重新整理已成功](#)
- [執行個體重新整理失敗](#)
- [已取消執行個體重新整理](#)
- [執行個體重新整理復原已啟動](#)
- [執行個體重新整理復原成功](#)
- [執行個體重新整理復原失敗](#)

達到檢查點

當已取代的執行個體數目達到針對檢查點定義的百分比閾值時，Amazon EC2 Auto Scaling 會傳送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Checkpoint Reached",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "ab00cf8f-9126-4f3c-8010-dbb8cad6fb86",
    "AutoScalingGroupName": "my-asg",
    "CheckpointPercentage": "50",
    "CheckpointDelay": "300"
  }
}
```

已啟動執行個體重新整理

在執行個體重新整理的狀態變更為 InProgress 時，Amazon EC2 Auto Scaling 會傳送以下事件。

```
{
```



```

"version": "0",
"id": "12345678-1234-1234-1234-123456789012",
"detail-type": "EC2 Auto Scaling Instance Refresh Started",
"source": "aws.autoscaling",
"account": "123456789012",
"time": "yyyy-mm-ddThh:mm:ssZ",
"region": "us-west-2",
"resources": [
  "auto-scaling-group-arn"
],
"detail": {
  "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
  "AutoScalingGroupName": "my-asg"
}
}

```

執行個體重新整理已成功

在執行個體重新整理的狀態變更為 Successful 時，Amazon EC2 Auto Scaling 會傳送以下事件。

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}

```

執行個體重新整理失敗

在執行個體重新整理的狀態變更為 Failed 時，Amazon EC2 Auto Scaling 會傳送以下事件。

```

{
  "version": "0",

```

```

{id": "12345678-1234-1234-1234-123456789012",
"detail-type": "EC2 Auto Scaling Instance Refresh Failed",
"source": "aws.autoscaling",
"account": "123456789012",
"time": "yyyy-mm-ddThh:mm:ssZ",
"region": "us-west-2",
"resources": [
  "auto-scaling-group-arn"
],
"detail": {
  "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
  "AutoScalingGroupName": "my-asg"
}
}

```

已取消執行個體重新整理

在執行個體重新整理的狀態變更為 Cancelled 時，Amazon EC2 Auto Scaling 會傳送以下事件。

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Cancelled",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}

```

執行個體重新整理復原已啟動

在執行個體重新整理的狀態變更為 RollbackInProgress 時，Amazon EC2 Auto Scaling 會傳送以下事件。

```

{
  "version": "0",

```

```
{
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Started",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

執行個體重新整理復原成功

在執行個體重新整理的狀態變更為 RollbackSuccessful 時，Amazon EC2 Auto Scaling 會傳送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

執行個體重新整理復原失敗

在執行個體重新整理的狀態變更為 Failed 時，Amazon EC2 Auto Scaling 會傳送以下事件。

```
{
  "version": "0",
```

```

    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Failed",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-west-2",
    "resources": [
      "auto-scaling-group-arn"
    ],
    "detail": {
      "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
      "AutoScalingGroupName": "my-asg"
    }
  }
}

```

執行個體重新整理取代根磁碟區生命週期事件

在將生命週期關聯新增至 Auto Scaling 群組時，Amazon EC2 Auto Scaling 會在執行個體轉換到等待狀態時，將事件傳送至 EventBridge。如需詳細資訊，請參閱[使用生命週期掛鉤取代根磁碟區](#)。

下列範例顯示當 Auto Scaling 群組的執行個體在執行個體重新整理期間正在進行根磁碟區取代時的事件。盡可能發出事件。

事件範例

- [開始取代根磁碟區生命週期動作](#)
- [結束取代根磁碟區生命週期動作](#)
- [事件模式範例](#)

開始取代根磁碟區生命週期動作

在執行個體重新整理期間選取執行個體進行根磁碟區取代時，其狀態會變更為 ReplacingRootVolume:Wait。

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",

```

```

"resources": [
  "auto-scaling-group-arn"
],
"detail": {
  "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
  "AutoScalingGroupName": "my-asg",
  "LifecycleHookName": "my-terminate-lifecycle-hook",
  "EC2InstanceId": "i-1234567890abcdef0",
  "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
  "NotificationMetadata": "additional-info",
  "Origin": "AutoScalingGroup",
  "Destination": "AutoScalingGroup",
  "Action": "ReplaceRootVolume"
}
}

```

結束取代根磁碟區生命週期動作

當執行個體完成根磁碟區取代，且其狀態變更為 時Pending:Wait。

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "AutoScalingGroup",
    "Action": "ReplaceRootVolume"
  }
}

```

事件模式範例

上一節提供 Amazon EC2 Auto Scaling 在執行個體重新整理期間以取代根磁碟區策略發出的範例事件。EventBridge 事件模式擁有與其相符事件相同的結構。該模式引用您欲比對的欄位，並提供您正在尋找的數值。事件中的下列欄位會形成規則中定義的事件模式，以呼叫動作：

```
"source": "aws.autoscaling"
```

識別該事件是來自 Amazon EC2 Auto Scaling。

```
"detail-type": "EC2 Instance-launch Lifecycle Action"
```

識別事件類型。

```
"Action": "ReplaceRootVolume"
```

識別在執行個體上執行的動作。

使用下列範例事件模式，擷取與準備取代其根磁碟區之執行個體相關聯的所有事件：

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ],
  "detail": {
    "Action": [ "ReplaceRootVolume" ]
  }
}
```

使用下列範例事件模式，擷取在完成根磁碟區取代後，與返回服務的執行個體相關聯的所有事件：

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "Action": [ "ReplaceRootVolume" ]
  }
}
```

暖集區範例事件和模式

Amazon EC2 Auto Scaling 支援 Amazon EventBridge 中的多種預先定義模式。這簡化了事件模式的建立方式。您在表單上選取欄位值，EventBridge 就會為您產生模式。目前，Amazon EC2 Auto

Scaling 不支援由具有暖集區的 Auto Scaling 群組發出之任何事件的預先定義模式。您輸入的模式必須以 JSON 物件的形式呈現。本節和 [為暖集區事件建立 EventBridge 規則](#) 主題向您示範如何使用事件模式來選取事件並將其傳送到目標。

若要建立可篩選 Amazon EC2 Auto Scaling 傳送至 EventBridge 的暖集區相關事件的 EventBridge 規則，請加入來自事件 detail 區段的 Origin 和 Destination 欄位。

Origin 和 Destination 的值可為以下項目：

EC2 | AutoScalingGroup | WarmPool

目錄

- [範例事件](#)
- [事件模式範例](#)

範例事件

在將生命週期關聯新增至 Auto Scaling 群組時，Amazon EC2 Auto Scaling 會在執行個體轉換到等待狀態時，將事件傳送至 EventBridge。如需詳細資訊，請參閱 [在 Auto Scaling 群組中使用具有暖集區的生命週期掛鉤](#)。

本節包括 Auto Scaling 群組具有暖集區時這些事件的範例。盡可能發出事件。

Note

如需 Amazon EC2 Auto Scaling 在擴展成功時傳送至 EventBridge 的事件，請參閱 [成功的擴展事件](#)。如需擴展不成功的事件，請參閱 [不成功的擴展事件](#)。

事件範例

- [橫向擴展生命週期動作](#)
- [縮減生命週期動作](#)

橫向擴展生命週期動作

在橫向擴展時將執行個體轉換到等待狀態時，交付的事件會以 EC2 Instance-launch Lifecycle Action 作為 detail-type 的值。在 detail 物件中，Origin 和 Destination 屬性的值顯示執行個體的來源和目的地。

在此範例橫向擴展事件中，新執行個體會啟動，且由於已新增至暖集區，因此其狀態會變更為 `Warmed:Pending:Wait`。如需詳細資訊，請參閱[暖集區中的執行個體生命週期狀態轉換](#)。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "WarmPool"
  }
}
```

在此範例橫向擴展事件中，由於執行個體已從暖集區新增至 Auto Scaling 群組，因此其狀態會變更為 `Pending:Wait`。如需詳細資訊，請參閱[暖集區中的執行個體生命週期狀態轉換](#)。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-19T00:35:52.359Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "19cc4d4a-e450-4d1c-b448-0de67EXAMPLE",
    "AutoScalingGroupName": "my-asg",
  }
```

```

    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "WarmPool",
    "Destination": "AutoScalingGroup"
  }
}

```

縮減生命週期動作

在縮減時將執行個體轉換到等待狀態時，交付的事件會以 EC2 Instance-terminate Lifecycle Action 作為 detail-type 的值。在 detail 物件中，Origin 和 Destination 屬性的值顯示執行個體的來源和目的地。

在此範例縮減事件中，由於執行個體已傳回至暖集區，因此其狀態會變更為 Warmed:Pending:Wait。如需詳細資訊，請參閱[暖集區中的執行個體生命週期狀態轉換](#)。

```

{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2022-03-28T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "42694b3d-4b70-6a62-8523-09a1eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-termination-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata": "additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "WarmPool"
  }
}

```

事件模式範例

前一節提供了由 Amazon EC2 Auto Scaling 發出的範例事件。

EventBridge 事件模式擁有與其相符事件相同的結構。該模式引用您欲比對的欄位，並提供您正在尋找的數值。

事件中的下列欄位會形成規則中定義的事件模式，以呼叫動作：

"source": "aws.autoscaling"

識別該事件是來自 Amazon EC2 Auto Scaling。

"detail-type": "*EC2 Instance-launch Lifecycle Action*"

識別事件類型。

"Origin": "*EC2*"

識別執行個體的來源。

"Destination": "*WarmPool*"

識別執行個體的目的地。

使用下列範例事件模式來擷取與進入暖集區的執行個體相關聯的所有 EC2 Instance-launch Lifecycle Action 事件。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "Origin": [ "EC2" ],
    "Destination": [ "WarmPool" ]
  }
}
```

使用下列範例事件模式來擷取與因橫向擴展事件而離開暖集區的執行個體相關聯的所有 EC2 Instance-launch Lifecycle Action 事件。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
```

```
    "Origin": [ "WarmPool" ],
    "Destination": [ "AutoScalingGroup" ]
  }
}
```

使用下列範例事件模式來擷取與直接啟動至 Auto Scaling 群組的執行個體相關聯的所有 EC2 Instance-launch Lifecycle Action 事件。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "Origin": [ "EC2" ],
    "Destination": [ "AutoScalingGroup" ]
  }
}
```

使用下列範例事件模式來擷取與在縮減時將執行個體傳回至暖集區相關聯的所有 EC2 Instance-terminate Lifecycle Action 事件。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ],
  "detail": {
    "Origin": [ "AutoScalingGroup" ],
    "Destination": [ "WarmPool" ]
  }
}
```

使用下列範例事件模式擷取與 EC2 Instance-launch Lifecycle Action 相關聯的所有事件，無論原始伺服器或目的地為何。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
}
```

使用 Amazon EventBridge 規則來自動化動作

當 Amazon EC2 Auto Scaling 發出事件時，事件通知會以 JSON 文件形式傳送至 Amazon EventBridge。您可以寫入 EventBridge 規則，以在事件模式符合規則時自動化要採取的動作。如果

EventBridge 偵測到事件模式符合規則中定義的模式，EventBridge 會叫用規則中指定的目標 (或多個目標)。

您可以使用本節中的範例程序做為起點。

下列文件可能也會提供您所需的協助。

- 若要使用 Lambda 函數在執行個體啟動或終止前對其執行自訂動作，請參閱 [教學課程：設定叫用 Lambda 函數的 lifecycle hook](#)。
- 若要在使用 CloudTrail 記錄的 API 呼叫叫用 Lambda 函數，請參閱《Amazon EventBridge 使用者指南》中的[教學課程：使用 EventBridge 記錄 AWS API 呼叫](#)。
- 如需有關建立事件規則的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的「[建立對事件做出反應的 Amazon EventBridge 規則](#)」。

主題

- [針對執行個體重新整理事件建立 EventBridge 規則](#)
- [為暖集區事件建立 EventBridge 規則](#)

針對執行個體重新整理事件建立 EventBridge 規則

下列範例建立傳送電子郵件通知的 EventBridge 規則。當執行個體重新整理期間到達檢查點時，系統會在 Auto Scaling 群組發出事件時執行此操作。其中包含使用 Amazon SNS 設定電子郵件通知的步驟。若要使用 Amazon SNS 傳送電子郵件通知，您必須先建立主題，然後以您的電子郵件地址訂閱此主題。

如需執行個體重新整理功能的詳細資訊，請參閱 [使用執行個體重新整理來更新 Auto Scaling 群組中的執行個體](#)。

建立 Amazon SNS 主題

SNS 主題是一種邏輯存取點，也是 Auto Scaling 群組用來傳送通知的通訊管道。您可以為您所建立的主題指定名稱。

主題名稱必須符合下列需求：

- 具有 1 到 256 個字元。
- 包含大小寫 ASCII 字母、數字、底線或連字號

如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的[建立 Amazon SNS 主題](#)。

訂閱 Amazon SNS 主題

若要收到 Auto Scaling 群組傳送到該主題的通知，您必須訂閱連結至主題的端點。在此過程中，請於 Endpoint (端點) 指定要用來接收 Amazon EC2 Auto Scaling 通知的電子郵件地址。

如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的[訂閱 Amazon SNS 主題](#)。

確認 Amazon SNS 訂閱

Amazon SNS 會將確認電子郵件傳送到您在先前步驟中所指定的電子郵件地址。

請務必從 AWS 通知開啟電子郵件，然後選擇連結以確認訂閱，然後再繼續下一個步驟。

您會收到來自 的確認訊息 AWS。Amazon SNS 現在已設定為採用您指定的電子郵件地址，以電子郵件方式接收和傳送通知。

將事件路由到 Amazon SNS 主題

建立符合所選事件的規則，並將其路由到 Amazon SNS 主題，以便通知訂閱的電子郵件地址。

若要建立向您的 Amazon SNS 主題傳送通知的規則

1. 前往 <https://console.aws.amazon.com/events/> 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇規則。
3. 選擇建立規則。
4. 針對 Define rule detail (定義規則詳細資訊) 執行下列動作：
 - a. 輸入規則的 Name (名稱)，或者輸入描述。

在同一個區域和同一個事件匯流排上，規則不能與另一個規則同名。
 - b. 針對事件匯流排選擇預設值。當您帳戶中 AWS 的服務產生事件時，一律會前往您帳戶的預設事件匯流排。
 - c. 針對規則類型，選擇具有事件模式的規則。
 - d. 選擇下一步。
5. 針對 Build event pattern (建置事件模式) 執行下列動作：

- a. 在 Event source (事件來源) 欄位中，選擇 AWS events or EventBridge partner events (事件或 EventBridge 合作夥伴事件)。
- b. 針對 Event pattern (事件模式)，請執行下列動作：
 - i. 在 Event source (事件來源)，選擇 AWS 服務。
 - ii. 對於 AWS 服務，選擇 Auto Scaling。
 - iii. 對於 Event type (事件類型)，請選擇 Instance Refresh (執行個體重新整理)。
 - iv. 在預設情況下，規則符合任何執行個體重新整理事件。若要建立規則，以便在執行個體重新整理期間到達檢查點時通知您，請選擇 Specific instance event(s) (特定執行個體事件)，然後選取 EC2 Auto Scaling Instance Refresh Checkpoint Reached (到達的 EC2 Auto Scaling 執行個體重新整理檢查點)。
 - v. 在預設情況下，規則符合區域中的任何 Auto Scaling 群組。為了讓規則符合特定 Auto Scaling 群組，請選擇 Specific group name(s) (特定群組名稱)，然後選取一或多個 Auto Scaling 群組。
 - vi. 選擇下一步。
6. 針對 Select target(s) (選取目標) 執行下列動作：
 - a. 對於 Target types (目標類型)，選擇 AWS 服務。
 - b. 對於 Select a target (選取目標)，選擇 SNS topic (SNS 主題)。
 - c. 對於 Topic (主題)，請選擇您的 Amazon SNS 主題。
 - d. (選用) 在 Additional settings (其他設定) 下，您可以選擇性地設定其他設定。如需詳細資訊，請參閱「Amazon EventBridge 使用者指南」中的[建立對事件做出反應的 Amazon EventBridge 規則](#)。
 - e. 選擇下一步。
7. (選用) 針對 Tags (標籤)，您可以選擇性地將一或多個標籤指派給您的規則，然後選擇 Next (下一步)。
8. 對於 Review and create (檢閱與建立)，請檢閱規則的詳細資訊，然後視需要來修改。然後，選擇 Create role (建立角色)。

為暖集區事件建立 EventBridge 規則

以下範例建立了 EventBridge 規則來呼叫程式設計動作。當新執行個體新增至暖集區時，系統會在 Auto Scaling 群組發出事件時執行此操作。

建立規則之前，請建立您希望規則用作目標的 AWS Lambda 函數。您必須將此函數指定為規則的目標。以下程序僅提供建立 EventBridge 規則 (這些規則會在新的執行個體進入暖集區時做出反應) 的步驟。如需說明如何建立簡單 Lambda 函數以便於傳入事件符合規則時叫用的簡介教學課程，請參閱 [教學課程：設定叫用 Lambda 函數的 lifecycle hook](#)。

如需如何建立與使用暖集區的詳細資訊，請參閱 [使用暖集區縮短開機時間的應用程式延遲](#)。

建立叫用 Lambda 函數的事件規則

1. 前往 <https://console.aws.amazon.com/events/> 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇規則。
3. 選擇建立規則。
4. 針對 Define rule detail (定義規則詳細資訊) 執行下列動作：
 - a. 輸入規則的 Name (名稱)，或者輸入描述。

在同一個區域和同一個事件匯流排上，規則不能與另一個規則同名。
 - b. 針對事件匯流排選擇預設值。當您 AWS 服務帳戶中的產生事件時，一律會前往您帳戶的預設事件匯流排。
 - c. 針對規則類型，選擇具有事件模式的規則。
 - d. 選擇下一步。
5. 針對 Build event pattern (建置事件模式) 執行下列動作：
 - a. 在 Event source (事件來源) 欄位中，選擇 AWS events or EventBridge partner events (事件或 EventBridge 合作夥伴事件)。
 - b. 對於 Event pattern (事件模式)，選擇 Custom pattern (JSON editor) (自訂模式 (JSON 編輯器))，並將下列模式貼入 Event pattern (事件模式) 方塊，把 ## 顯示文字取代為您的 Auto Scaling 群組名稱。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
    "AutoScalingGroupName": [ "my-asg" ],
    "Origin": [ "EC2" ],
    "Destination": [ "WarmPool" ]
  }
}
```


若要建立符合其他事件的規則，請修改事件模式。如需詳細資訊，請參閱[事件模式範例](#)。

- c. 選擇下一步。
6. 針對 Select target(s) (選取目標) 執行下列動作：
 - a. 對於 Target types (目標類型)，選擇 AWS 服務。
 - b. 對於 Select a target (選取目標)，選擇 Lambda function (Lambda 函數)。
 - c. 在 Function (函數) 欄位中，請選擇您要將事件傳送至哪個函數。
 - d. (選用) 對於 Configure version/alias (設定版本/別名)，輸入目標 Lambda 函數的版本和別名設定。
 - e. (選用) 對於 Additional settings (其他設定)，輸入適合您應用程式的其他設定。如需詳細資訊，請參閱「Amazon EventBridge 使用者指南」中的[建立對事件做出反應的 Amazon EventBridge 規則](#)。
 - f. 選擇下一步。
7. (選用) 針對 Tags (標籤)，您可以選擇性地將一或多個標籤指派給您的規則，然後選擇 Next (下一步)。
8. 對於 Review and create (檢閱與建立)，請檢閱規則的詳細資訊，然後視需要來修改。然後，選擇 Create role (建立角色)。

使用 Amazon VPC 為您的 Auto Scaling 執行個體提供網際網路連線

Amazon Virtual Private Cloud (Amazon VPC) 是一項服務，可讓您在您定義的邏輯隔離虛擬網路中啟動 AWS 資源，例如 Auto Scaling 群組。

在 Amazon VPC 中的子網路是由 VPC IP 地址範圍內的某個區段所定義的可用區域內的分支。使用子網路時，您可以根據安全性和操作需求分組執行個體。子網路整個在所建立的可用區域內。您在子網路中啟動 Auto Scaling 執行個體。

若要啟用子網路中的網際網路和執行個體之間的通訊，您必須建立網際網路閘道，並將它連接到您的 VPC。網際網路閘道可讓子網路內的資源透過 Amazon EC2 網路節點連線至網際網路。若子網路的流量路由至網際網路閘道，則該子網路便稱為公有子網路。若子網路的流量未路由至網際網路閘道，則該子網路便稱為私有子網路。針對必須連結至網際網路的資源使用公有子網路，並針對不需連結至網際網路的資源使用私有子網路。如需有關在 VPC 中對執行個體授予網際網路存取權的詳細資訊，請參閱《Amazon VPC 使用者指南》中的[存取網際網路](#)。

目錄

- [預設 VPC](#)
- [非預設 VPC](#)
- [選擇 VPC 子網路時的注意事項](#)
- [VPC 中的 IP 地址指定](#)
- [VPC 中的網路介面](#)
- [執行個體安置租用](#)
- [AWS Outposts](#)
- [了解 VPC 的更多資源](#)

預設 VPC

如果您在 2013 年 12 月 4 日 AWS 帳戶 之後建立，或者您正在新的 中建立 Auto Scaling 群組 AWS 區域，我們會為您建立預設 VPC。您的預設 VPC 隨附在每個可用區域的預設子網路中。如果已有預設 VPC，您的 Auto Scaling 群組依預設會建立在預設的 VPC 中。

您可以在 Amazon VPC 主控台的[您的 VPC 頁面](#)上檢視您的 VPC。

如需預設 VPC 的詳細資訊，請參閱《Amazon VPC 使用者指南》中的[預設 VPC](#)。

非預設 VPC

您可以選擇建立其他 VPC，方法是前往 AWS 管理主控台 中的 [VPC 儀表板頁面](#)，然後選取 Create VPC (建立 VPC)。

如需詳細資訊，請參閱《[Amazon VPC 使用者指南](#)》。

Note

VPC 遍及整個 AWS 區域內的所有可用區域。向 VPC 新增子網時，請選擇多個可用區域，以確保這些子網中託管的應用程式具有高可用性。可用區域是一個或多個獨立的資料中心，具備 AWS 區域中的備援電源、聯網和連線能力。可用區域會協助您使生產應用程式具備高可用性、容錯能力和可擴展性。

選擇 VPC 子網路時的注意事項

為 Auto Scaling 群組選擇 VPC 子網時請注意以下事項：

- 若要將 Elastic Load Balancing 負載平衡器連接到 Auto Scaling 群組，則可將執行個體啟動到公有或私有子網路中。不過，負載平衡器只能在公有子網路中建立以支援 DNS 解析。
- 如果直接透過 SSH 存取 Auto Scaling 執行個體，則只能將執行個體啟動到公有子網路中。
- 如果您使用 AWS Systems Manager Session Manager 存取無輸入 Auto Scaling 執行個體，則可以在公有或私有子網路中啟動執行個體。
- 如果使用私有子網路，則可以允許 Auto Scaling 執行個體使用公有 NAT 閘道存取網際網路。
- 依預設，預設 VPC 中的預設子網路為公有子網路。

VPC 中的 IP 地址指定

在 VPC 中啟動 Auto Scaling 執行個體時，會從該執行個體啟動的子網路 CIDR 範圍中自動指派一個私有 IP 地址給您的執行個體。這可以讓您的執行個體與 VPC 中的其他執行個體通訊。

您可以設定啟動範本或啟動組態，以指派公有 IPv4 地址給您的執行個體。將公有 IP 地址指派給您的執行個體，可讓它們與網際網路或其他 AWS 服務通訊。

當您將執行個體啟動到已設定的子網路來自動指派 IPv6 地址時，它們會收到 IPv4 和 IPv6 地址。否則，它們只會收到 IPv4 地址。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [IPv6 地址](#)。

如需為 VPC 或子網路指定 CIDR 範圍的詳細資訊，請參閱《[Amazon VPC 使用者指南](#)》。

當您使用可指定其他網路介面的啟動範本時，Amazon EC2 Auto Scaling 可以在執行個體啟動時自動指派其他私有 IP 地址。每個網路介面都會指派一個單一私有 IP 地址，而此地址來自該執行個體啟動所在的子網路 CIDR 範圍中。在這種情況下，系統就無法再自動指派公有 IPv4 地址給主要網路介面。除非您將可用彈性 IP 地址與 Auto Scaling 執行個體建立關聯，否則將無法透過公有 IPv4 地址連線至執行個體。

VPC 中的網路介面

VPC 中的每個執行個體都有預設的網路介面 (主要網路介面)。您無法分離主要網路介面和執行個體。您可以建立額外的網路介面，然後連接到您 VPC 中的任一執行個體。您可連接的網路介面數會隨執行個體類型而不同。

使用啟動範本啟動執行個體時，您可以指定其他網路介面。不過，啟動具有多個網路介面的 Auto Scaling 執行個體，會自動在與執行個體相同的子網路中建立每個介面。這是因為 Amazon EC2 Auto Scaling 會忽略啟動範本中定義的子網路，以便支持 Auto Scaling 群組中指定的子網路。如需詳細資訊，請參閱[建立 Auto Scaling 群組的啟動範本](#)。

如果將相同子網路中兩個或以上的網路介面建立到或連接到執行個體，您可能會遇到聯網問題，例如非對稱路由，特別是在使用非 Amazon Linux 變體的執行個體之上。如果需要這種類型的組態，您必須在作業系統內設定次要網路介面。如需範例，請參閱 AWS 知識中心的[如何讓次要網路介面在 Ubuntu EC2 執行個體中運作？](#)。

執行個體安置租用

根據預設，VPC 中的所有執行個體都會以共用租用執行個體執行。Amazon EC2 Auto Scaling 也支援專用執行個體和專用主機。如需詳細資訊，請參閱[使用進階設定建立啟動範本](#)。

AWS Outposts

AWS Outposts 使用區域中可存取的 VPC 元件，包括網際網路閘道、虛擬私有閘道、Amazon VPC Transit Gateways 和 VPC 端點，將 Amazon VPC 從區域擴展 AWS 到 Outpost。Outpost 位於區域中的可用區域，且為該可用區域的延伸，可用於復原。

如需詳細資訊，請參閱「[AWS Outposts 使用者指南](#)」。

如需瞭解如何部署從 Outpost 內 Application Load Balancer 轉發流量的 Auto Scaling 群組，請參閱以下部落格文章在[AWS Outposts 上設定 Application Load Balancer](#)。

了解 VPC 的更多資源

若要進一步了解 VPC 與子網路，請參閱下列主題。

- 在 VPC 中的私有子網路
 - [範例：在私有子網路和 NAT 具有伺服器的 VPC](#)
 - [NAT 閘道](#)
- 在 VPC 中的公有子網路
 - [範例：測試環境的 VPC](#)
 - [範例：適用於 Web 和資料庫伺服器的 VPC](#)
- Application Load Balancer 的子網路
 - [負載平衡器的子網路](#)
- 一般 VPC 資訊
 - [《Amazon VPC 使用者指南》](#)
 - [使用 VPC 對等互連來連線 VPC](#)
 - [彈性網路介面](#)

- [使用 VPC 端點進行私有連線](#)

Amazon EC2 Auto Scaling 中的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為符合最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[共同責任模式](#)將其描述為雲端的安全性，和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也為您提供可安全使用的服務。在[AWS 合規計劃](#)中，第三方稽核人員會定期測試和驗證我們的安全有效性。若要了解適用於 Amazon EC2 Auto Scaling 的合規計劃，請參閱[AWS 合規計劃範圍內的服務](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您的公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 Amazon EC2 Auto Scaling 時套用共同責任模型。下列各主題將說明如何配置 Amazon EC2 Auto Scaling 來達成您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon EC2 Auto Scaling 資源。

主題

- [Amazon EC2 Auto Scaling 中的基礎設施安全](#)
- [Amazon EC2 Auto Scaling 中的彈性](#)
- [Amazon EC2 Auto Scaling 中的資料保護](#)
- [Amazon EC2 Auto Scaling 的 Identity and Access Management](#)
- [Amazon EC2 Auto Scaling 的合規驗證](#)
- [Amazon EC2 Auto Scaling 和介面 VPC 端點](#)

Amazon EC2 Auto Scaling 中的基礎設施安全

Amazon EC2 Auto Scaling 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及 如何 AWS 保護基礎設施的相關資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取 Amazon EC2 Auto Scaling。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。

- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

您也可以將虛擬私有雲端 (VPC) 端點用於 Amazon EC2 Auto Scaling。介面 VPC 端點可讓 Amazon VPC 資源使用其私有 IP 地址來存取 Amazon EC2 Auto Scaling，而不會暴露於公有網際網路。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 和介面 VPC 端點](#)

相關資源

如需隔離 Amazon EC2 提供服務流量的功能資訊，請參閱《[Amazon EC2 使用者指南](#)》中的 [Amazon EC2 中的基礎設施安全](#)。Amazon EC2

Amazon EC2 Auto Scaling 中的彈性

AWS 全球基礎設施是以 AWS 區域 和 可用區域為基礎建置。AWS 區域 提供多個實體隔離和隔離的可用區域，這些可用區域以低延遲、高輸送量和高度備援聯網連接。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

若要受益於可用區域設計的地理備援，請執行下列動作：

- 讓您的 Auto Scaling 群組跨越多個可用區域。
- 在每個可用區域內維持至少一個執行個體。
- 連接負載平衡器，將傳入流量分配到相同的可用區域。如果您使用 Application Load Balancer，則請啟用跨區域負載平衡，以確保每個 EC2 執行個體獲得相似的流量。這有助於在容錯移轉事件期間限制負載增加對現有執行個體所造成的影響，並比不使用跨區域負載平衡產生更大的復原能力。
- 請確保已正確設定 Elastic Load Balancing 運作狀態檢查，並且已在 Auto Scaling 群組上啟用這些檢查。然後，如果執行個體的運作狀態檢查失敗，則 Elastic Load Balancing 會停止向其傳送流量，並將流量重新路由到運作狀態良好的執行個體，同時 Amazon EC2 Auto Scaling 取代運作狀態不佳的執行個體。

Amazon EC2 Auto Scaling 以下列方式協助支援您的應用程式復原需求：

- 檢查執行個體是否存在運作狀態和連線問題。當執行個體運作狀態不佳時，它會自動終止該執行個體，並啟動新的執行個體。

- 如果動態擴展政策生效，則會根據傳入流量自動擴展容量。
- 偵測支援擴展政策的 Amazon CloudWatch 指標可靠性問題，並在無法使用可靠的指標 (例如資料點遺失) 時暫停縮減活動。
- 在群組擴展時，嘗試在每個已啟用的可用區域中維持同等數量的執行個體。
- 使用可用區域維持高可用性。當可用區域運作狀態不佳時，Amazon EC2 Auto Scaling 會執行下列動作：
 - 在為 Auto Scaling 群組啟用的不同可用區域中啟動新執行個體。
 - 當運作狀態不佳的可用區域恢復到良好運作狀態時，將執行個體重新分佈到所有啟用的可用區域中。
- 如果執行個體無法在指定的可用區域中啟動，則請繼續嘗試在其他已啟用的可用區域中啟動執行個體。
- 自動向您的 Auto Scaling 群組所關聯的負載平衡器註冊和取消註冊執行個體。這樣，您就不需要單獨註冊和取消註冊執行個體。
- Amazon EC2 Auto Scaling 服務 APIs 的控制平面中斷不會影響現有 Auto Scaling 群組的擴展。

相關資源

如需協助支援 Amazon EBS 提供之資料彈性需求的功能資訊，請參閱《Amazon EBS 使用者指南》[中的 Amazon Elastic Block Store 中的彈性](#)。

Amazon EC2 Auto Scaling 中的資料保護

AWS [共同責任模型](#)適用於 Amazon EC2 Auto Scaling 中的資料保護。如此模型所述，AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱[資料隱私權常見問答集](#)。如需有關歐洲資料保護的相關資訊，請參閱AWS 安全性部落格上的[AWS 共同責任模型和 GDPR](#) 部落格文章。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。

- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《使用者指南》中的[使用 CloudTrail 追蹤](#)。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱[聯邦資訊處理標準 \(FIPS\) 140-3](#)。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 Amazon EC2 Auto Scaling 或使用 AWS 服務 主控台、API AWS CLI或其他 AWS SDKs 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

當您啟動 Amazon EC2 執行個體時，您可以選擇將使用者資料傳遞給執行個體，以在執行個體開機時執行其他組態。我們也建議您絕不要在將要傳送到執行個體的使用者資料中加入機密或敏感資訊。

使用 AWS KMS keys 加密 Amazon EBS 磁碟區

您可以設定 Auto Scaling 群組，以此加密透過 AWS KMS keys 存放於雲端的 Amazon EBS 磁碟區資料。Amazon EC2 Auto Scaling 支援 AWS 受管金鑰和客戶受管金鑰來加密您的資料。請注意，使用啟動組態時，無法使用 KmsKeyId 選項來指定客戶受管金鑰。若要指定客戶受管金鑰，請改用啟動範本。如需詳細資訊，請參閱[為 Auto Scaling 群組建立啟動範本](#)。如需有關如何建立、存放和管理 AWS KMS 加密金鑰的資訊，請參閱 [AWS Key Management Service 開發人員指南](#)。

您既可以在 EBS 後端 AMI 中設定客戶受管金鑰，然後再設定啟動範本或啟動組態；也可以使用預設加密功能，強制加密您建立的新 EBS 磁碟區和快照副本。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[使用加密搭配 EBS 支援的 AMIs](#)，以及《Amazon EBS 使用者指南》中的[預設加密](#)。

Amazon EC2

Note

如需如何設定在使用客戶受管金鑰進行加密時，啟動 Auto Scaling 執行個體所需之金鑰政策的相關資訊，請參閱 [與加密磁碟區搭配使用的必要 AWS KMS 金鑰政策](#)。

相關資源

如需 Amazon EBS 提供的資料保護指導方針，請參閱《Amazon EBS [使用者指南](#)》中的 [Amazon Elastic Block Store 中的資料保護](#)。

與加密磁碟區搭配使用的必要 AWS KMS 金鑰政策

Amazon EC2 Auto Scaling [使用服務連結角色](#) 將許可委派給其他 AWS 服務。Amazon EC2 Auto Scaling 服務連結角色已預先定義，並包含 Amazon EC2 Auto Scaling AWS 服務 代表您呼叫其他 所需的許可。預先定義的許可也包含對 的存取 AWS 受管金鑰。不過，其不包括對客戶受管金鑰的存取權，該存取權允許您保有對這些金鑰的完整控制權。

本主題介紹在為 Amazon EBS 加密指定客戶受管金鑰時，如何設定啟動 Auto Scaling 執行個體所需的金鑰政策。

Note

Amazon EC2 Auto Scaling 不需要額外的授權，就能使用預設的 AWS 受管金鑰 來保護帳戶中的加密磁碟區。

目錄

- [概要](#)
- [設定金鑰政策](#)
- [範例 1：允許存取客戶受管金鑰的金鑰政策區段](#)
- [範例 2：允許跨帳戶存取客戶受管金鑰的金鑰政策區段](#)
- [在 AWS KMS 主控台中編輯金鑰政策](#)

概要

當 Amazon EC2 Auto Scaling 啟動執行個體時，AWS KMS keys 以下可用於 Amazon EBS 加密：Amazon EC2

- [AWS 受管金鑰](#) – Amazon EBS 在您的帳戶中建立、擁有和管理的加密金鑰。這是新帳戶的預設加密金鑰。除非您指定客戶受管金鑰，否則 AWS 受管金鑰 會用於加密。
- [客戶受管金鑰](#) – 您所建立、擁有和管理的自訂加密金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [建立金鑰](#)。

注意：金鑰必須為對稱金鑰。Amazon EBS 不支援非對稱客戶受管金鑰。

您可以在建立會指定加密磁碟區的加密快照或啟動範本時，設定客戶受管金鑰，或依預設啟用加密。

設定金鑰政策

您的 KMS 金鑰必須具有金鑰政策，允許 Amazon EC2 Auto Scaling 使用透過客戶受管金鑰加密的 Amazon EBS 磁碟區來啟動執行個體。

請使用此頁面上的範例來設定金鑰政策，為客戶受管金鑰提供 Amazon EC2 Auto Scaling 存取權。您可以在建立金鑰時修改客戶受管金鑰的金鑰政策，或之後再修改。

您必須至少新增兩個政策陳述式到金鑰政策，該政策才能與 Amazon EC2 Auto Scaling 搭配運作。

- 第一個陳述式允許 Principal 元素中指定的 IAM 身分直接使用客戶受管金鑰。它包含對金鑰執行 AWS KMS Encrypt、Decrypt、GenerateDataKey*、ReEncrypt* 和 DescribeKey 操作的許可。
- 第二個陳述式允許 Principal 元素中指定的 IAM 身分使用 CreateGrant 操作來產生授予，將其自身許可的子集委派給與 AWS KMS 或其他主體整合 AWS 服務的。這允許其使用金鑰來代表您建立加密資源。

在您為金鑰政策新增政策陳述式時，請勿變更政策中任何現有的陳述式。

對於下列每個範例，必須取代的引數，例如金鑰 ID 或服務連結角色的名稱，會顯示為 #####。在大多數情況下，您可以將服務連結角色的名稱取代為 Amazon EC2 Auto Scaling 服務連結角色的名稱。

如需詳細資訊，請參閱下列資源：

- 若要使用 建立金鑰 AWS CLI，請參閱 [create-key](#)。
- 若要使用 更新金鑰政策 AWS CLI，請參閱 [put-key-policy](#)。
- 若要尋找金鑰 ID 和 Amazon Resource Name (ARN)，請參閱《AWS Key Management Service 開發人員指南》中的 [尋找金鑰 ID 和 ARN](#)。
- 如需 Amazon EC2 Auto Scaling 服務連結角色的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 的服務連結角色](#)。
- 如需有關 Amazon EBS 加密和 KMS 一般的資訊，請參閱《[Amazon EBS 使用者指南](#)》和《[開發人員指南](#)》中的 [Amazon EBS 加密](#)。 [AWS Key Management Service](#)

範例 1：允許存取客戶受管金鑰的金鑰政策區段

請將以下兩個政策陳述式新增至客戶受管金鑰的金鑰政策，以將範例 ARN 取代為可存取金鑰的適當服務連結角色的 ARN。在此範例中，該政策區段會授予服務連結角色名為 `AWSServiceRoleForAutoScaling` 的許可，讓他們能夠使用客戶受管金鑰。

```
{
  "Sid": "Allow service-linked role use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

```
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
    ]
  },
  "Action": [
    "kms:CreateGrant"
  ],
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:GrantIsForAWSResource": true
    }
  }
}
```

```
}
```

範例 2：允許跨帳戶存取客戶受管金鑰的金鑰政策區段

如果您建立的客戶受管金鑰所在帳戶與 Auto Scaling 群組不同，您必須使用搭配金鑰政策的授權，以便允許存取跨帳戶存取權金鑰。

必須依照下列順序完成兩個步驟：

1. 首先，將以下兩個政策陳述式新增至客戶受管金鑰的金鑰政策。將範例 ARN 取代為其他帳戶的 ARN，請務必將 **111122223333** 取代為 AWS 帳戶您要建立 Auto Scaling 群組的 的實際帳戶 ID。這允許您為特定帳戶中的 IAM 使用者或角色授予許可，以使用下文的 CLI 命令為金鑰建立授權。但是，這不會將任何使用者存取權提供給金鑰。

```
{
  "Sid": "Allow external account 111122223333 use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:root"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

```
{
  "Sid": "Allow attachment of persistent resources in external
account 111122223333",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:root"
    ]
  },
  "Action": [
```

```

    "kms:CreateGrant"
  ],
  "Resource": "*"
}

```

2. 然後，從您要建立 Auto Scaling 群組的所在帳戶建立授權以將相關許可委派給適當的服務連結角色。授權的 Grantee Principal 元素是適當服務連結角色的 ARN。key-id 為金鑰的 ARN。

以下是 [create-grant](#) CLI 命令範例，此命令可對帳戶 **111122223333** 中名為 `AWSServiceRoleForAutoScaling` 的服務連結角色，提供在 **444455556666** 帳戶中使用客戶受管金鑰的許可。

```

aws kms create-grant \
  --region us-west-2 \
  --key-id arn:aws:kms:us-west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \
  --grantee-principal arn:aws:iam::111122223333:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling \
  --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey" "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"

```

若要讓此命令成功，提出請求的使用者必須有執行 `CreateGrant` 動作的許可。

下列 IAM 政策範例允許帳戶 **111122223333** 中的 IAM 身分 (使用者或角色)，為帳戶 **444455556666** 中的客戶受管金鑰建立許可。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreationOfGrantForTheKMSKeyinExternalAccount444455556666",
      "Effect": "Allow",
      "Action": "kms:CreateGrant",
      "Resource": "arn:aws:kms:us-west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    }
  ]
}

```

如需在其他 AWS 帳戶中為 KMS 金鑰建立授權的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的「[AWS KMS 中的授權](#)」。

 Important

指定為承授者主體的服務連結角色名稱必須是現有角色的名稱。建立授與後，為確保授與允許 Amazon EC2 Auto Scaling 使用指定的 KMS 金鑰，請勿刪除和重新建立服務連結角色。

在 AWS KMS 主控台中編輯金鑰政策

前幾節的範例僅顯示如何對金鑰政策新增陳述式，而這只是其中一個用來變更金鑰政策的方法。變更金鑰政策的最簡單方法是針對金鑰政策使用 AWS KMS 主控台的預設檢視，並將 IAM 身分（使用者或角色）設為適當金鑰政策的其中一個金鑰使用者。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[使用 AWS 管理主控台 預設檢視](#)。

 Important

請務必謹慎進行。主控台的預設檢視政策陳述式包含對客戶受管金鑰執行 AWS KMS Revoke 操作的許可。如果您授予帳戶中客戶受管金鑰的 AWS 帳戶存取權，而且您不小心撤銷授予他們此許可的授予，外部使用者將無法再存取其加密的資料或用來加密其資料的金鑰。

Amazon EC2 Auto Scaling 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一種 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可以控制驗證 (登入) 和授權 (具有許可) 來使用 Amazon EC2 Auto Scaling 資源。IAM 是您可以免費使用 AWS 服務的。

若要使用 Amazon EC2 Auto Scaling，您需要 AWS 帳戶和安全登入資料才能登入您的帳戶。如需詳細資訊，請參閱《IAM 使用者指南》中的[AWS 安全登入](#)資料。

如需完整的 IAM 文件，請參閱 [IAM 使用者指南](#)。

存取控制

您可以持有效憑證來對請求進行身分驗證，但還須具備許可，才能建立或存取 Amazon EC2 Auto Scaling 資源。例如，您必須有建立 Auto Scaling 群組、使用啟動範本來啟動執行個體等的許可。

下列各節詳細說明 IAM 管理員如何藉由控制可執行 Amazon EC2 Auto Scaling 動作的人員，使用 IAM 以協助保護您的 Amazon EC2 Auto Scaling 資源。

我們建議您先閱讀 Amazon EC2 主題。請參閱《[Amazon EC2 使用者指南](#)》中的 [Amazon EC2 的 Identity and Access Management](#)。Amazon EC2 在閱讀本節中的主題之後，您應該清楚了解 Amazon EC2 提供哪些存取控制許可，以及這些許可如何配合您的 Amazon EC2 Auto Scaling 資源使用權限。

主題

- [Amazon EC2 Auto Scaling 與 IAM 搭配運作的方式](#)
- [Amazon EC2 Auto Scaling 的 API 許可](#)
- [AWS Amazon EC2 Auto Scaling 的 受管政策](#)
- [Amazon EC2 Auto Scaling 的服務連結角色](#)
- [Amazon EC2 Auto Scaling 身分型政策範例](#)
- [預防跨服務混淆代理人](#)
- [在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量](#)
- [在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)

Amazon EC2 Auto Scaling 與 IAM 搭配運作的方式

在您使用 IAM 以管理對 Amazon EC2 Auto Scaling 的存取權之前，了解哪些 IAM 功能可以與 Amazon EC2 Auto Scaling 搭配使用。

可以與 Amazon EC2 Auto Scaling 搭配使用的 IAM 功能

IAM 功能	Amazon EC2 Auto Scaling 支援
身分型政策	是
資源型政策	否
政策動作	是

IAM 功能	Amazon EC2 Auto Scaling 支援
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
服務角色	是
服務連結角色	是

若要全面了解 Amazon EC2 Auto Scaling 和其他 如何與大多數 IAM 功能 AWS 服務 搭配使用，請參閱 [《AWS 服務 IAM 使用者指南》中的 與 IAM 搭配使用](#)。

Amazon EC2 Auto Scaling 的身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 [IAM JSON 政策元素參考](#)。

Amazon EC2 Auto Scaling 內的資源型政策

支援資源型政策：否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中[指定主體](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以在其他帳戶內指定所有帳戶或 IAM 實體作為資源型政策的主體。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 中的快帳戶資源存取](#)。

Amazon EC2 Auto Scaling 的政策動作

支援政策動作：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。政策會使用動作來授予執行相關聯動作的許可。

如要查看 Amazon EC2 Auto Scaling 動作清單，請參閱《服務授權參考》中的 [Amazon EC2 Auto Scaling 定義的動作](#)。

Amazon EC2 Auto Scaling 中的政策動作會在動作之前使用以下字首：

```
autoscaling
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [  
  "autoscaling:action1",  
  "autoscaling:action2"  
]
```

可以使用萬用字元 (*) 指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "autoscaling:Describe*"
```

Amazon EC2 Auto Scaling 的政策資源

支援政策資源：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

```
"Resource": "*"
```

您可以使用 ARN 來識別 IAM 政策適用的 Auto Scaling 群組和和啟動組態。

Auto Scaling 群組具有以下 ARN。

```
"Resource": "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
```

啟動組態具有以下 ARN。

```
"Resource": "arn:aws:autoscaling:region:account-id:launchConfiguration:uuid:launchConfigurationName/lc-name"
```

若要使用 CreateAutoScalingGroup 動作指定 Auto Scaling 群組，您必須將 UUID 取代為萬用字元 (*)，如以下範例所示。

```
"Resource": "arn:aws:autoscaling:region:account-id:autoScalingGroup:*:autoScalingGroupName/asg-name"
```

若要使用 CreateLaunchConfiguration 動作指定啟動組態，您必須將 UUID 取代為萬用字元 (*)，如以下範例所示。

```
"Resource": "arn:aws:autoscaling:region:account-id:launchConfiguration:*:launchConfigurationName/lc-name"
```

如需有關 Amazon EC2 Auto Scaling 資源類型及其 ARN 的詳細資訊，請參閱《服務授權參考》中的 [由 Amazon EC2 Auto Scaling 定義的資源](#)。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 [Amazon EC2 Auto Scaling 定義的動作](#)。

Note

如需使用 ARN 控制對 Auto Scaling 群組的存取權的 IAM 政策範例，請參閱 [控制哪些為可刪除的 Auto Scaling 群組](#)。

並非所有 Amazon EC2 Auto Scaling 動作皆支援資源層級的許可。對於不支援資源層級許可的動作，您必須使用萬用字元 (*) 作為資源。

以下 Amazon EC2 Auto Scaling 動作皆不支援資源層級的許可。

- DescribeAccountLimits
- DescribeAdjustmentTypes
- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeAutoScalingNotificationTypes
- DescribeInstanceRefreshes
- DescribeLaunchConfigurations
- DescribeLifecycleHooks
- DescribeLifecycleHookTypes
- DescribeLoadBalancers
- DescribeLoadBalancerTargetGroups
- DescribeMetricCollectionTypes
- DescribeNotificationConfigurations
- DescribePolicies
- DescribeScalingActivities
- DescribeScalingProcessTypes
- DescribeScheduledActions
- DescribeTags
- DescribeTerminationPolicyTypes
- DescribeWarmPool

Amazon EC2 Auto Scaling 的政策條件金鑰

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素會根據定義的條件，指定陳述式的執行時機。您可以建立使用[條件運算子](#)的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的[AWS 全域條件內容索引鍵](#)。

Amazon EC2 Auto Scaling 支援下列條件索引鍵，可用於控制支援動作的存取權和強制執行 Auto Scaling 群組的設定：

- `autoscaling:InstanceTypes`
- `autoscaling:LaunchConfigurationName`
- `autoscaling:LaunchTemplateVersionSpecified`
- `autoscaling:LoadBalancerNames`
- `autoscaling:MaxSize`
- `autoscaling:MinSize`
- `autoscaling:ResourceTag/key-name: tag-value`
- `autoscaling:TargetGroupARNs`
- `autoscaling:VPCZoneIdentifiers`
- `autoscaling:ForceDelete`

以下條件金鑰特定於建立啟動組態請求：

- `autoscaling:ImageId`
- `autoscaling:InstanceType`
- `autoscaling:MetadataHttpEndpoint`
- `autoscaling:MetadataHttpPutResponseHopLimit`
- `autoscaling:MetadataHttpTokens`
- `autoscaling:SpotPrice`

Amazon EC2 Auto Scaling 也支援下列全域條件金鑰，您可以使用這些金鑰根據請求中的標籤或 Auto Scaling 群組中的標籤來定義許可。如需詳細資訊，請參閱[標記 Auto Scaling 群組和執行個體](#)。

- `aws:RequestTag/key-name: tag-value`
- `aws:ResourceTag/key-name: tag-value`
- `aws:TagKeys: [tag-key, ...]`

如要了解哪些 Amazon EC2 Auto Scaling 動作可與條件金鑰搭配使用，請參閱《服務授權參考》中的[Amazon EC2 Auto Scaling 定義的動作](#)。如需有關 Amazon EC2 Auto Scaling 條件索引鍵的詳細資訊，請參閱[Amazon EC2 Auto Scaling 的條件索引鍵](#)。

Note

如需使用條件索引鍵來控制支援動作存取權和強制執行 Auto Scaling 群組的設定的 IAM 政策範例，請參閱下列資源：

- [需要啟動範本和版本編號](#) – 此範例會強制在建立或更新 Auto Scaling 群組時，必須指定啟動範本和啟動範本的版本編號。
- [控制可以建立的 Auto Scaling 群組大小](#) – 此範例會在建立或更新具有特定標籤的 Auto Scaling 群組時，對 MinSize 和 MaxSize 屬性的可能值強制執行限制。
- [控制哪些為可刪除的擴展政策](#) – 此範例會強制僅允許刪除沒有特定標籤的 Auto Scaling 群組的擴展政策。

Amazon EC2 Auto Scaling 中的 ACL

支援 ACL：否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

ABAC 與 Amazon EC2 Auto Scaling

支援 ABAC (政策中的標籤)：部分

屬性型存取控制 (ABAC) 是一種授權策略，根據稱為標籤的屬性定義許可權。您可以將標籤連接至 IAM 實體 AWS 和資源，然後設計 ABAC 政策，以便在委託人的標籤符合資源上的標籤時允許操作。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的 [條件元素](#) 中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的 [使用 ABAC 授權定義許可](#)。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的 [使用屬性型存取控制 \(ABAC\)](#)。

ABAC 可用於支援標籤的資源，但並非所有支援標籤的資源。啟動組態和擴展政策不支援標籤，但 Auto Scaling 群組支援標籤。

如需詳細資訊，請參閱 [標記 Auto Scaling 群組和執行個體](#)。

透過 Amazon EC2 Auto Scaling 使用臨時憑證

支援臨時憑證：是

臨時登入資料提供對 AWS 資源的短期存取，並在您使用聯合或切換角色時自動建立。AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 中的臨時安全憑證與可與 IAM 搭配運作的 AWS 服務](#)。

Amazon EC2 Auto Scaling 的服務角色

支援服務角色：是

服務角色是服務擔任的 [IAM 角色](#)，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可給 AWS 服務](#)。

當您建立會通知 Amazon SNS 主題或 Amazon SQS 佇列的 lifecycle hook 時，您必須指定一個角色，以允許 Amazon EC2 Auto Scaling 代表您存取 Amazon SNS 或 Amazon SQS。使用 IAM 主控台來設定 lifecycle hook 的服務角色。主控台可協助您使用受管政策建立具有足夠許可集的角色。如需詳細資訊，請參閱[使用 Amazon SNS 接收通知](#)及[使用 Amazon SQS 接收通知](#)。

建立 Auto Scaling 群組時，您可以選擇性地傳入服務角色，以允許 Amazon EC2 執行個體 AWS 服務代表您存取其他執行個體。Amazon EC2 執行個體的服務角色 (也稱為啟動範本或啟動組態的 Amazon EC2 執行個體設定檔) 是一種特殊服務角色類型，當執行個體啟動時，該角色被指派給 Auto Scaling 群組中的每個 EC2 執行個體。您可以使用 IAM 主控台和 AWS CLI 來建立或編輯此服務角色。如需詳細資訊，請參閱[在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)。

Warning

變更服務角色的許可可能會中斷 Amazon EC2 Auto Scaling 功能。只有在 Amazon EC2 Auto Scaling 提供指引時，才能編輯服務角色。

Amazon EC2 Auto Scaling 的服務連結角色

支援服務連結角色：是

服務連結角色是連結至的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 AWS 帳戶，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需有關建立或管理 Amazon EC2 Auto Scaling 服務連結角色的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 的服務連結角色](#)。

Amazon EC2 Auto Scaling 的 API 許可

您必須將許可授予給使用者，以呼叫他們所需的 Amazon EC2 Auto Scaling API 動作，如 [Amazon EC2 Auto Scaling 的政策動作](#) 中所述。此外，對於某些 Amazon EC2 Auto Scaling 動作，您必須授予使用者從其他 AWS APIs 許可。

從其他 AWS APIs 取得的必要許可

除了 Amazon EC2 Auto Scaling API 許可之外，使用者還必須擁有來自其他 AWS APIs 下列許可，才能成功執行相關聯的動作。

建立 Auto Scaling 群組 (autoscaling:CreateAutoScalingGroup)

- iam:CreateServiceLinkedRole – 建立預設服務連結角色，如果該角色尚未存在。
- iam:PassRole – 在啟動時將 IAM 角色傳遞至 服務或 EC2 執行個體。提供非預設服務連結角色、lifecycle hook 的 IAM 角色或指定執行個體設定檔 (IAM 角色的容器) 的啟動範本時，則需要此項目。
- ec2:RunInstances – 在提供啟動範本時啟動執行個體。
- ec2:CreateTags – 在提供具有標籤規格的啟動範本時，在啟動時標記執行個體和磁碟區。

建立 lifecycle hook (autoscaling:PutLifecycleHook)

- iam:PassRole – 將 IAM 角色傳遞至 服務。提供 IAM 角色時需要。

連接 VPC Lattice 目標群組 (autoscaling:AttachTrafficSources)

- vpc-lattice:RegisterTargets – 自動向目標群組註冊執行個體。

分離 VPC Lattice 目標群組 (autoscaling:DetachTrafficSources)

- vpc-lattice:DeregisterTargets – 使用目標群組自動取消註冊執行個體。

建立啟動組態 (autoscaling:CreateLaunchConfiguration)

- ec2:DescribeImages
- ec2:DescribeInstances
- ec2:DescribeInstanceAttribute
- ec2:DescribeKeyPairs
- ec2:DescribeSecurityGroups

- `ec2:DescribeSpotInstanceRequests`
- `ec2:DescribeVpcClassicLink`
- `iam:PassRole` – 在啟動時將 IAM 角色傳遞至 EC2 執行個體。啟動組態指定執行個體設定檔 (IAM 角色的容器) 時需要此項目。

AWS Amazon EC2 Auto Scaling 的 受管政策

AWS 受管政策是由 AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您可以開始將許可指派給使用者、群組和角色。

請記住，AWS 受管政策可能不會授予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義特定於使用案例的[客戶管理政策](#)，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新受 AWS 管政策中定義的許可，則更新會影響政策連接的所有委託人身分（使用者、群組和角色）。AWS 服務當新的 啟動或新的 API 操作可用於現有服務時，AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱 IAM 使用者指南中的 [AWS 受管政策](#)。

Amazon EC2 Auto Scaling 受管政策

您可以將下列受管政策連接至您的 AWS Identity and Access Management (IAM) 身分（使用者或角色）。每個政策會提供對 Amazon EC2 Auto Scaling 所有或部分 API 動作的存取權。

- [AutoScalingConsoleFullAccess](#) – 使用 授予 Amazon EC2 Auto Scaling 的完整存取權 AWS 管理主控台。在您使用啟動組態時適用此政策，但在您使用啟動範本時不適用。
- [AutoScalingConsoleReadOnlyAccess](#) – 授予使用 對 Amazon EC2 Auto Scaling 的唯讀存取權 AWS 管理主控台。在您使用啟動組態時適用此政策，但在您使用啟動範本時不適用。
- [AutoScalingFullAccess](#) – 針對需要從 AWS CLI 或 SDK 完整存取 Amazon EC2 Auto Scaling 但無法存取的 IAM 身分，授予 Amazon EC2 Auto Scaling 的完整 AWS 管理主控台 存取權。SDKs
- [AutoScalingReadOnlyAccess](#) – 針對僅呼叫 AWS CLI 或 SDK 的 IAM 身分，授予 Amazon EC2 Auto Scaling 的唯讀存取權。SDKs

當您從主控台使用啟動範本時，需要授予啟動範本特定的額外許可，這會在 [在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量](#) 中予以討論。Amazon EC2 Auto Scaling 主控台需要 `ec2` 動作的許可，以便顯示啟動範本和使用啟動範本啟動執行個體的相關資訊。

AutoScalingServiceRolePolicy AWS 受管政策

此政策會連接到服務連結角色，允許 Amazon EC2 Auto Scaling 代表您執行動作。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 的服務連結角色](#)。

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的 [AutoScalingServiceRolePolicy](#)。

AWS 受管政策的 Amazon EC2 Auto Scaling 更新

檢視自此服務開始追蹤 Amazon EC2 Auto Scaling AWS 受管政策更新以來的詳細資訊。如需有關此頁面變更的自動提醒，請訂閱 Amazon EC2 Auto Scaling 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	Date
Amazon EC2 Auto Scaling 會將許可更新為其服務連結角色	此AutoScalingService RolePolicy 政策現在包含 ec2:CreateReplaceRootVolumeTask 許可，可讓您在執行個體重新整理時取代執行個體上的根磁碟區。此外，events:DescribeRule 許可已從政策中移除。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的服務連結角色 。	2025 年 11 月 18 日
Amazon EC2 Auto Scaling 將許可新增至其服務連結角色	AutoScalingService RolePolicy 政策現在包含呼叫 AWS Resource Groups ListGroupResources API 動作的許可，以取得屬於指定資源群組成員的所有資源名稱 (ARNs)。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的服務連結角色 。	2024 年 11 月 20 日
Amazon EC2 Auto Scaling 將許可新增至其服務連結角色	AutoScalingService RolePolicy 政策現在授予	2024 年 2 月 29 日

變更	描述	Date
	呼叫 Amazon EC2 GetSecurityGroupsForVpc API 動作的許可，以取得 VPC 的所有安全群組以改善驗證，以及 Amazon EC2 GetInstanceTypesFromInstanceRequirements API 動作，以取得哪些執行個體類型符合特定執行個體要求的相關資訊。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的服務連結角色 。	

變更	描述	Date
Amazon EC2 Auto Scaling 將許可新增至其服務連結角色	AutoScalingService RolePolicy 政策現在將許可授予給該服務，以存取與 VPC Lattice 整合所需的 API 動作。 • GetTargetGroup 和 ListTargetGroup 動作。需要擷取有關 VPC Lattice 目標群組的資訊。 • RegisterTargets 和 DeregisterTargets 動作。需要從 VPC Lattice 目標群組註冊和取消註冊執行個體。 • ListTargets 。允許 Amazon EC2 Auto Scaling 擷取已註冊到 VPC Lattice 目標群組的執行個體的運作狀態資訊。 如需詳細資訊，請參閱Amazon EC2 Auto Scaling 的服務連結角色。	2022 年 12 月 6 日

變更	描述	Date
Amazon EC2 Auto Scaling 將許可新增至其服務連結角色	若要在建立啟動範本時支援使用 AWS Systems Manager 參數做為 AMI ID 的別名，AutoScalingService RolePolicy 政策現在會授予呼叫 AWS Systems Manager GetParameters API 動作的許可。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的服務連結角色 。	2022 年 3 月 28 日
Amazon EC2 Auto Scaling 將許可新增至其服務連結角色	為了支援預測擴展，AutoScalingService RolePolicy 政策現已包含呼叫 CloudWatch GetMetricData API 動作的許可。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的服務連結角色 。	2021 年 5 月 19 日
Amazon EC2 Auto Scaling 已開始追蹤變更	Amazon EC2 Auto Scaling 開始追蹤其 AWS 受管政策的變更。	2021 年 5 月 19 日

Amazon EC2 Auto Scaling 的服務連結角色

Amazon EC2 Auto Scaling 使用服務連結角色，來獲得代表您呼叫其他 AWS 服務 所需的許可。服務連結角色是直接連結至 的唯一 IAM 角色類型 AWS 服務。

服務連結角色提供安全的方式對其他 AWS 服務 委派許可，因為只有連結的服務能夠擔任服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立服務連結角色](#)。服務連結角色也可透過 顯示所有 API 呼叫 AWS CloudTrail。因為這可讓您追蹤 Amazon EC2 Auto Scaling 代表您執行的所有動作，所以有助於監控和稽核要求。如需詳細資訊，請參閱 [使用 記錄 Amazon EC2 Auto Scaling API 呼叫 AWS CloudTrail](#)。

以下幾節描述如何建立和管理 Amazon EC2 Auto Scaling 服務連結角色。一開始請先設定許可，以允許 IAM 身分 (例如使用者或角色) 可以建立、編輯或刪除服務連結角色。

目錄

- [概要](#)
- [服務連結角色授予的許可](#)
- [Amazon EC2 Auto Scaling 服務連結角色的支援區域](#)
- [建立、編輯和刪除服務連結角色](#)
 - [建立服務連結角色 \(自動\)](#)
 - [建立服務連結角色 \(手動\)](#)
 - [編輯服務連結角色](#)
 - [刪除服務連結角色](#)

概要

Amazon EC2 Auto Scaling 服務連結角色有兩種類型：

- 適用於帳戶的預設服務連結角色，名稱為 `AWSServiceRoleForAutoScaling`。系統會自動將此角色指派至您的 Auto Scaling 群組 (除非您指定不同的服務連結角色)。
- 服務連結角色，具有您在建立角色時指定的自訂尾碼，例如 `AWSServiceRoleForAutoScaling_mysuffix`。

自訂字尾服務連結角色的許可與預設服務連結角色的許可完全相同。但不論是哪種角色，您都無法編輯，而且也無法在仍由 Auto Scaling 群組使用中的情況下加以刪除。兩者的差別只在角色名稱的字尾。

您可以在編輯 AWS Key Management Service 金鑰政策時指定任一角色，以允許使用客戶受管金鑰加密由 Amazon EC2 Auto Scaling 啟動的執行個體。不過，如果您打算對特定的客戶受管金鑰提供精密存取權時，請使用自訂字尾服務連結角色。使用自訂字尾服務連結角色可讓您：

- 提升對客戶受管金鑰的控制力
- 在 CloudTrail 日誌中追蹤有哪些 Auto Scaling 群組發出 API 呼叫

如果您建立客戶受管金鑰，但不想讓所有使用者皆可存取，請依照以下步驟來允許自訂字尾服務連結角色的使用：

1. 使用自訂字尾建立服務連結角色。如需詳細資訊，請參閱 [建立服務連結角色 \(手動\)](#)。
2. 對服務連結角色授予客戶受管金鑰的存取權。如需如何讓金鑰政策允許服務連結角色使用金鑰的詳細資訊，請參閱 [與加密磁碟區搭配使用的必要 AWS KMS 金鑰政策](#)。
3. 允許使用者存取您建立的服務連結角色。如需建立 IAM 政策的詳細資訊，請參閱 [控制可傳遞哪些服務連結角色 \(使用 PassRole\)](#)。如果使用者嘗試在未經許可的情況下，指定服務連結角色將該角色傳遞給服務，他們會收到錯誤。

服務連結角色授予的許可

Amazon EC2 Auto Scaling 使用名為 `AWSServiceRoleForAutoScaling` 的服務連結角色或您的自訂尾碼服務連結角色。

服務連結角色信任下列服務來擔任此角色：

- `autoscaling.amazonaws.com`

角色許可政策 [AutoScalingServiceRolePolicy](#) 可讓 Amazon EC2 Auto Scaling 完成下列動作：

- `ec2` – 建立、描述、修改、啟動/停止和終止 EC2 執行個體。
- `iam` – 將 [IAM 角色傳遞](#) 至 EC2 執行個體，讓執行個體上執行的應用程式可以存取角色的臨時登入資料。
- `iam` – 建立 `AWSServiceRoleForEC2Spot` 服務連結角色，以允許 Amazon EC2 Auto Scaling 代表您啟動 Spot 執行個體。
- `elasticloadbalancing` – 向 Elastic Load Balancing 註冊和取消註冊執行個體，並檢查已註冊目標的運作狀態。
- `cloudwatch` – 建立、描述、修改和刪除用於擴展政策的 CloudWatch 警示，並擷取用於預測擴展的指標。
- `sns` – 執行個體啟動或終止時，將通知發佈至 Amazon SNS。
- `events` – 代表您建立、描述、更新和刪除 EventBridge 規則。
- `ssm` – 使用 Systems Manager 參數做為啟動範本中 AMI ID 的別名時，從參數存放區讀取參數。
- `vpc-lattice` – 向 VPC Lattice 註冊和取消註冊執行個體，並檢查已註冊目標的運作狀態。
- `resource-groups` – 取得屬於指定資源群組成員之資源的所有資源名稱 (ARNs)。

Amazon EC2 Auto Scaling 服務連結角色的支援區域

Amazon EC2 Auto Scaling 支援 AWS 區域 在提供服務的所有 中使用服務連結角色。

建立、編輯和刪除服務連結角色

建立服務連結角色 (自動)

除非手動建立自訂字尾服務連結角色，並在建立群組時予以指定，否則 Amazon EC2 Auto Scaling 會在首次建立 Auto Scaling 群組時，為您建立 AWSServiceRoleForAutoScaling 服務連結角色。

您必須具備 IAM 許可才能建立服務連結角色。否則自動建立會失敗。如需詳細資訊，請參閱《IAM 使用者指南》中的[服務連結角色許可](#)，以及本指南中的[建立服務連結角色](#)。

建立服務連結角色 (手動)

建立服務連結角色 (主控台)

1. 前往網址 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色)、Create role (建立新角色)。
3. 對於 Select trusted entity (選取信任的實體) 區段，選擇 AWS service (AWS 服務)。
4. 針對 Choose the service that will use this role (選擇將使用此角色的服務)，選擇 EC2 Auto Scaling 和 EC2 Auto Scaling 使用案例。
5. 選擇 Next: Permissions (下一步：許可)、Next: Tags (下一步：標籤) 及 Next: Review (下一步：檢閱)。注意：您不能在建立角色時，將標籤連接到服務連結角色。
6. 在 Review (檢閱) 頁面上，讓 Role name (角色名稱) 保持空白以建立名為 AWSServiceRoleForAutoScaling 的服務連結角色，或輸入字尾來建立名為 AWSServiceRoleForAutoScaling_## 的服務連結角色。
7. (選用) 針對 Role description (角色描述)，編輯服務連結角色的描述。
8. 選擇建立角色。

建立服務連結角色 (AWS CLI)

使用下列 [create-service-linked-role](#) CLI 命令，為名為 AWSServiceRoleForAutoScaling_*suffix* 的 Amazon EC2 Auto Scaling 建立服務連結角色。

```
aws iam create-service-linked-role --aws-service-name autoscaling.amazonaws.com --  
custom-suffix suffix
```


此命令的輸出包含服務連結角色的 ARN，您可以使用它來授予服務連結角色存取客戶受管金鑰權限。

```
{
  "Role": {
    "RoleId": "ABCDEF0123456789ABCDEF",
    "CreateDate": "2018-08-30T21:59:18Z",
    "RoleName": "AWSServiceRoleForAutoScaling_suffix",
    "Arn": "arn:aws:iam::123456789012:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_suffix",
    "Path": "/aws-service-role/autoscaling.amazonaws.com/",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": [
            "sts:AssumeRole"
          ],
          "Principal": {
            "Service": [
              "autoscaling.amazonaws.com"
            ]
          },
          "Effect": "Allow"
        }
      ]
    }
  }
}
```

如需詳細資訊，請參閱《IAM 使用者指南》中的[建立服務連結角色](#)。

編輯服務連結角色

您不能編輯針對 Amazon EC2 Auto Scaling 所建立的服務連結角色。在建立服務連結角色後，就無法變更角色的名稱或其許可。不過，您可以編輯角色的描述。如需詳細資訊，請參閱 IAM 使用者指南中的[編輯服務連結角色描述](#)。

刪除服務連結角色

如果您不會使用 Auto Scaling 群組，建議您刪除其服務連結角色。刪除該角色後，您就不會有未使用或主動監控和維護的實體。

您必須先刪除相關的相依資源，才能刪除服務連結角色。這可避免您意外撤銷 Amazon EC2 Auto Scaling 使用資源的許可。如果某個服務連結角色與多個 Auto Scaling 群組搭配使用，您必須先刪除使用該服務連結角色的所有 Auto Scaling 群組，才能刪除該角色。如需詳細資訊，請參閱 [刪除 Auto Scaling 基礎設施](#)。

您可以使用 IAM 來刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的 [刪除服務連結角色](#)。

在刪除 AWSServiceRoleForAutoScaling 服務連結角色之後，Amazon EC2 Auto Scaling 會在您建立 Auto Scaling 群組時再次建立角色，但不指定不同的服務連結角色。

Amazon EC2 Auto Scaling 身分型政策範例

根據預設，您 中的新使用者 AWS 帳戶 沒有執行任何動作的許可。IAM 管理員必須建立和指派 IAM 政策，它們可提供 IAM 身分 (例如使用者或角色) 許可，以執行 Amazon EC2 Auto Scaling API 動作。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 政策，請參閱《IAM 使用者指南》中的 [在 JSON 標籤上建立政策](#)。

以下顯示許可政策範例。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:UpdateAutoScalingGroup",
      "autoscaling>DeleteAutoScalingGroup"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
    }
  },
  {
    "Effect": "Allow",
    "Action": "autoscaling:Describe*",
    "Resource": "*"
  }
}
```

```
}]
}
```

此政策範例會授與建立、更新和刪除 Auto Scaling 群組的許可，但前提是群組使用標籤 **purpose=testing**。由於 Describe 動作不支援資源層級許可，因此無需任何條件，但您必須在不同的陳述式中指定它們。若要使用啟動範本啟動執行個體，使用者也必須擁有 `ec2:RunInstances` 許可。如需詳細資訊，請參閱[在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量](#)。

Note

您可以建立自己的自訂 IAM 政策，以允許或拒絕 IAM 身分 (使用者或角色) 執行 Amazon EC2 Auto Scaling 動作的許可。您可以將這些自訂政策連接至需要指定許可的 IAM 身分。以下範例示範一些常用案例的許可。

有些 Amazon EC2 Auto Scaling API 動作允許您在政策中包含可由動作建立或修改的特定 Auto Scaling 群組。您可以指定個別 Auto Scaling 群組 ARN 來限制這些動作的目標資源。不過，最佳實務是建議您使用標籤型政策，允許 (或拒絕) 對有特定標籤的 Auto Scaling 群組執行動作。

範例

- [控制可以建立的 Auto Scaling 群組大小](#)
- [控制可以使用哪些標籤鍵和標籤值](#)
- [控制哪些為可刪除的 Auto Scaling 群組](#)
- [控制哪些為可刪除的擴展政策](#)
- [控制執行個體重新整理動作的存取權](#)
- [建立服務連結角色](#)
- [控制可傳遞哪些服務連結角色 \(使用 PassRole\)](#)

控制可以建立的 Auto Scaling 群組大小

下列政策會授與許可，以建立和更新有標籤 **environment=development** 的所有 Auto Scaling 群組的許可，只要請求者不指定下限小於 **1** 或上限大於 **10** 的大小即可。盡可能使用標籤來協助控制帳戶中 Auto Scaling 群組的存取權。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:UpdateAutoScalingGroup"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": { "autoscaling:ResourceTag/environment":
"development" },
      "NumericGreaterThanEqualsIfExists": { "autoscaling:MinSize": 1 },
      "NumericLessThanEqualsIfExists": { "autoscaling:MaxSize": 10 }
    }
  }]
}
```

或者，如果您不是使用標籤來控制 Auto Scaling 群組的存取權，您可以使用 ARN 來識別 IAM 政策適用的 Auto Scaling 群組。

Auto Scaling 群組具有以下 ARN。

```
"Resource": "arn:aws:autoscaling:region:account-id:autoScalingGroup:*:autoScalingGroupName/my-asg"
```

您也可以將多個 ARN 包含在清單中來加以指定。如需有關在 Resource 元素中指定 Amazon EC2 Auto Scaling 資源的 ARN 的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 的政策資源](#)。

控制可以使用哪些標籤鍵和標籤值

您也可以在 IAM 政策中使用條件來控制可套用至 Auto Scaling 群組的標籤鍵和標籤值。若要授與建立或標記 Auto Scaling 群組的許可，只要請求者指定特定的標籤，則可使用 `aws:RequestTag` 條件金鑰。若要僅允許特定的標籤鍵，請使用 `aws:TagKeys` 條件索引鍵和 `ForAllValues` 修飾詞。

下列政策需要請求者在請求中使用 **environment** 金鑰來指定標籤。`"?*" 值會針對標籤鍵強制執行一些值。若要使用萬用字元時，您必須使用 StringLike 條件運算子。`

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:CreateOrUpdateTags"
    ],
    "Resource": "*",
    "Condition": {
      "StringLike": { "aws:RequestTag/environment": "?*" }
    }
  }]
}
```

下列政策指定請求者只能使用標籤 **purpose=webserver** 和標籤 **cost-center=cc123** 來標記 Auto Scaling 群組，並且只允許 **purpose** 和 **cost-center** 標籤 (無法指定其他標籤)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:CreateOrUpdateTags"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/purpose": "webserver",
        "aws:RequestTag/cost-center": "cc123"
      },
      "ForAllValues:StringEquals": { "aws:TagKeys": ["purpose", "cost-center"] }
    }
  }]
}
```

```
}
```

下列政策需要請求者在請求中指定至少一個標籤，並僅允許 **cost-center** 和 **owner** 金鑰。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:CreateOrUpdateTags"
    ],
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": { "aws:TagKeys": ["cost-center", "owner"] }
    }
  }]
}
```

Note

以條件而言，條件索引鍵不區分大小寫，而條件值會區分大小寫。因此，欲強制標籤鍵區分大小寫，請使用 `aws:TagKeys` 條件索引鍵，其中標籤鍵指定為條件值。

控制哪些為可刪除的 Auto Scaling 群組

只有在群組有標籤 **environment=development** 時，下列政策才允許刪除 Auto Scaling 群組。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "autoscaling:DeleteAutoScalingGroup",
```

```

    "Resource": "*",
    "Condition": {
        "StringEquals": { "aws:ResourceTag/environment": "development" }
    }
}

```

下列政策使用 `autoscaling:ForceDelete` 條件金鑰來控制對 `DeleteAutoScalingGroup` API 動作的存取。這可防止某些使用者使用強制刪除操作，這會在刪除群組時終止 Auto Scaling 群組中的所有執行個體。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": "autoscaling:DeleteAutoScalingGroup",
    "Resource": "*",
    "Condition": {
      "Bool": {
        "autoscaling:ForceDelete": "true"
      }
    }
  }]
}

```

或者，如果您不使用條件金鑰來控制 Auto Scaling 群組的存取權，則可以指定 `Resource` 元素中資源的 ARN 來控制存取權。

下列政策提供使用者使用 `DeleteAutoScalingGroup` API 動作的許可，但僅適用於名稱開頭為的 Auto Scaling 群組 `devteam-`。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {

```

```

        "Effect": "Allow",
        "Action": "autoscaling:DeleteAutoScalingGroup",
        "Resource": "arn:aws:autoscaling:us-east-1:111122223333:autoScalingGroup:*:autoScalingGroupName/devteam-*"
    }
]
}

```

您也可以將多個 ARN 包含在清單中來加以指定。包含 UUID 以確保將存取權授予特定 Auto Scaling 群組。新群組的 UUID 與同名但已刪除群組的 UUID 不同。

```

"Resource": [
    "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-1",
    "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-2",
    "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/devteam-3"
]

```

控制哪些為可刪除的擴展政策

下列政策會授與許可，以使用 DeletePolicy 動作刪除擴展政策。但是，如果被採取行動的 Auto Scaling 群組具有標籤 **environment=production**，其也會拒絕該動作。盡可能使用標籤來協助控制帳戶中 Auto Scaling 群組的存取權。

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "autoscaling:DeletePolicy",
        "Resource": "*"
    },
    {
        "Effect": "Deny",
        "Action": "autoscaling:DeletePolicy",
        "Resource": "*",
        "Condition": {

```



```

        "StringEquals": { "autoscaling:ResourceTag/environment": "production" }
      }
    }]
  }

```

控制執行個體重新整理動作的存取權

下列政策僅在執行的 Auto Scaling 群組具有標籤 **environment=testing** 時，才會授與啟動、復原和取消執行個體重新整理的權限。由於 Describe 動作不支援資源層級許可，因此無需任何條件，但您必須在不同的陳述式中指定它們。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "autoscaling:StartInstanceRefresh",
      "autoscaling:CancelInstanceRefresh",
      "autoscaling:RollbackInstanceRefresh"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": { "autoscaling:ResourceTag/environment": "testing" }
    }
  },
  {
    "Effect": "Allow",
    "Action": "autoscaling:DescribeInstanceRefreshes",
    "Resource": "*"
  }
]}

```

若要在 StartInstanceRefresh 呼叫中指定所需的組態，使用者可能需要相關許可，例如：

- ec2 : RunInstances – 若要使用啟動範本啟動 EC2 執行個體，使用者必須在 IAM 政策中擁有 ec2:RunInstances 許可。如需詳細資訊，請參閱 [在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量](#)。

- `ec2:CreateTags` – 若要從在建立時將標籤新增至執行個體和磁碟區的啟動範本啟動 EC2 執行個體，使用者必須在 IAM 政策中擁有 `ec2:CreateTags` 許可。如需詳細資訊，請參閱[標記執行個體和磁碟區所需的許可](#)。
- `iam:PassRole` – 若要從包含執行個體描述檔 (IAM 角色的容器) 的啟動範本啟動 EC2 執行個體，使用者也必須擁有 IAM 政策中的 `iam:PassRole` 許可。如需詳細資訊和 IAM 政策範例，請參閱[在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)。
- `ssm:GetParameters` – 若要從使用 AWS Systems Manager 參數的啟動範本啟動 EC2 執行個體，使用者也必須擁有 IAM 政策中的 `ssm:GetParameters` 許可。如需詳細資訊，請參閱[在啟動範本中使用 AWS Systems Manager 參數而非 AMI IDs](#)。

建立服務連結角色

Amazon EC2 Auto Scaling 需要許可，才能在第一次呼叫 Amazon EC2 Auto Scaling API AWS 帳戶動作時建立服務連結角色。如果服務連結角色不存在，Amazon EC2 Auto Scaling 會在您的帳戶中建立該角色。服務連結角色會授予 Amazon EC2 Auto Scaling 的許可，以便可以 AWS 服務 代表您呼叫其他。

為能成功自動建立該角色，使用者必須已獲許可執行 `iam:CreateServiceLinkedRole` 動作。

```
"Action": "iam:CreateServiceLinkedRole"
```

以下顯示的是許可政策範例，即允許使用者為 Amazon EC2 Auto Scaling 建立 Amazon EC2 Auto Scaling 服務連結角色。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
    "Condition": {
      "StringLike": { "iam:AWSServiceName": "autoscaling.amazonaws.com" }
    }
  }]
}
```

控制可傳遞哪些服務連結角色 (使用 PassRole)

如果使用者建立或更新 Auto Scaling 群組，並在請求中指定自訂字尾服務連結角色，則需要 `iam:PassRole` 許可。

如果您授予不同服務連結角色存取不同金鑰的權限，您可以使用 `iam:PassRole` 許可來保護 AWS KMS 客戶受管金鑰的安全性。根據您的組織需求，您可能有一個金鑰適用於開發團隊、另一個適用於 QA 團隊，還有一個則適用於財務團隊。首先，建立一個可存取所需金鑰的服務連結角色，例如名為 `AWSServiceRoleForAutoScaling_devteamkeyaccess` 的服務連結角色。然後，將該政策連接至 IAM 身分，例如使用者或角色。

下列政策會授與許可，以將 `AWSServiceRoleForAutoScaling_devteamkeyaccess` 角色傳遞至名稱開頭為 `devteam-` 的任何 Auto Scaling 群組。如果建立 Auto Scaling 群組的 IAM 身分嘗試指定不同的服務連結角色，則會收到錯誤。如果他們選擇不指定服務連結角色，則會改用預設 `AWSServiceRoleForAutoScaling` 角色。

如需自訂字尾服務連結角色的詳細資訊，請參閱 [Amazon EC2 Auto Scaling 的服務連結角色](#)。

預防跨服務混淆代理人

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。

在中 AWS，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。

為了防止這種情況，AWS 提供工具，協助您保護所有服務的資料，其服務主體已獲得您帳戶中資源的存取權。建議使用 Amazon EC2 Auto Scaling 服務角色的信任政策中的 [aws:SourceArn](#) 和 [aws:SourceAccount](#) 全域條件內容金鑰。這些金鑰會限制 Amazon EC2 Auto Scaling 為資源提供其他服務的許可。

當 Amazon EC2 Auto Scaling 使用 AWS Security Token Service (AWS STS) 代表您擔任角色時，會設定 `SourceArn` 和 `SourceAccount` 欄位的值。

若要使用 `aws:SourceArn` 或 `aws:SourceAccount` 全域條件金鑰，請將值設定為 Amazon EC2 Auto Scaling 所儲存資源的 Amazon Resource Name (ARN) 或帳戶。盡可能使用 `aws:SourceArn`，這更為具體。對於 ARN 的未知部分，將值設定為 ARN 或具有萬用字元 (*) 的 ARN 模式。如果您不知道資源的 ARN，請改為使用 `aws:SourceAccount`。

下列範例示範如何使用 Amazon EC2 Auto Scaling 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容金鑰，來預防混淆代理人問題。

範例：使用 `aws:SourceArn` 和 `aws:SourceAccount` 條件金鑰

服務會擔任代您執行動作的角色稱為[服務角色](#)。如果您想要建立會將通知傳送到 Amazon EventBridge 以外的任何位置的 lifecycle hook，您必須建立服務角色，以允許 Amazon EC2 Auto Scaling 代表您將通知傳送到 Amazon SNS 主題或 Amazon SQS 佇列。如果您想要僅允許一個 Auto Scaling 群組與跨服務存取相關聯，則可以按如下方式指定服務角色的信任政策。

此信任政策範例使用條件陳述式，將服務角色的 AssumeRole 能力限制為僅影響指定帳戶中指定 Auto Scaling 群組的動作。獨立評估 `aws:SourceArn` 和 `aws:SourceAccount` 條件。使用服務角色的任何請求都必須滿足這兩個條件。

在使用此政策之前，請將區域、帳戶 ID、UUID 和群組名稱取代為您帳戶的有效值。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConfusedDeputyPreventionExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "autoscaling.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:autoscaling:us-east-1:111122223333:autoScalingGroup:uuid:autoScalingGroupName/my-asg"
        },
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

在上述範例中：

- Principal 元素會指定服務 (autoscaling.amazonaws.com) 的服務主體。
- Action 元素指定 sts:AssumeRole 動作。
- Condition 元素指定 aws:SourceArn 和 aws:SourceAccount 全域條件金鑰。來源的 ARN 包括帳戶 ID，因此不需要搭配 aws:SourceAccount 使用 aws:SourceArn。

其他資訊

如需詳細資訊，請參閱《IAM 使用者指南》中的[AWS 全域條件內容金鑰](#)、[混淆代理人問題](#)和[更新角色信任政策](#)。

在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量

Amazon EC2 Auto Scaling 支援將 Amazon EC2 啟動範本與您的 Auto Scaling 群組搭配使用。我們建議您允許使用者從啟動範本建立 Auto Scaling 群組，因為這樣做可讓使用者使用 Amazon EC2 Auto Scaling 和 Amazon EC2 的最新功能。例如，使用者必須指定啟動範本，才能使用[混合執行個體政策](#)。

您可以使用該 AmazonEC2FullAccess 政策授予使用者在其帳戶中使用 Amazon EC2 Auto Scaling 資源、啟動範本和其他 EC2 資源的完整存取權。您也可以建立自訂 IAM 政策，授予使用者精細許可可以與啟動範本搭配使用，如本主題所述。

您可以針對自身使用量身打造的範例政策

以下內容顯示您可以針對自身使用量身打造的基本許可政策範例。此政策會授與建立、更新和刪除所有 Auto Scaling 群組的許可，但前提是群組使用標籤 **purpose=testing**。接著，授與所有 Describe 動作的許可。由於 Describe 動作不支援資源層級許可，因此無需任何條件，但您必須在不同的陳述式中指定它們。

使用此政策的 IAM 身分 (使用者或角色) 擁有使用啟動範本建立或更新 Auto Scaling 群組的許可，因為其亦獲得使用 ec2:RunInstances 動作的許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```

        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling>DeleteAutoScalingGroup"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "autoscaling:Describe*",
        "ec2:RunInstances"
    ],
    "Resource": "*"
}
]
}

```

建立或更新 Auto Scaling 群組的使用者可能需要一些相關許可，例如：

- `ec2:CreateTags` – 若要在建立時將標籤新增至執行個體和磁碟區，使用者必須在 IAM 政策中擁有 `ec2:CreateTags` 許可。如需詳細資訊，請參閱 [標記執行個體和磁碟區所需的許可](#)。
- `iam:PassRole` – 若要從包含執行個體描述檔 (IAM 角色的容器) 的啟動範本啟動 EC2 執行個體，使用者也必須擁有 IAM 政策中的 `iam:PassRole` 許可。如需詳細資訊和 IAM 政策範例，請參閱 [在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色](#)。
- `ssm:GetParameters` – 若要從使用 AWS Systems Manager 參數的啟動範本啟動 EC2 執行個體，使用者也必須擁有 IAM 政策中的 `ssm:GetParameters` 許可。如需詳細資訊，請參閱 [在啟動範本中使用 AWS Systems Manager 參數而非 AMI IDs](#)。

當使用者與 Auto Scaling 群組互動時，會檢查啟動執行個體時要完成動作的這些許可。如需詳細資訊，請參閱 [ec2:RunInstances](#) 和 [iam:PassRole](#) 的許可驗證。

以下範例顯示您可用來控制 IAM 使用者需要使用啟動範本時具有之存取權限的政策陳述式。

主題

- [需要具有特定標籤的啟動範本](#)
- [需要啟動範本和版本編號](#)

- [需要使用執行個體中繼資料服務版本第 2 版 \(IMDSv2\)](#)
- [限制對 Amazon EC2 資源的存取](#)
- [標記執行個體和磁碟區所需的許可](#)
- [其他啟動範本許可](#)
- [ec2:RunInstances 和 iam:PassRole 的許可驗證](#)
- [相關資源](#)

需要具有特定標籤的啟動範本

授予 `ec2:RunInstances` 許可時，您可以指定使用者只能使用具有特定標籤或特定 ID 的啟動範本，以限制使用啟動範本啟動執行個體時的許可。您也可以控制 AMI 和其他資源，讓使用啟動範本的任何人都可以在啟動執行個體時參照和使用，方法是指定 `RunInstances` 呼叫的其他資源層級許可。

以下範例會限制使用位於指定區域且具有 **`purpose=testing`** 標籤的啟動範本來執行 `ec2:RunInstances` 動作的許可。它也可讓使用者存取啟動範本中指定的資源：AMI、執行個體類型、磁碟區、金鑰對、網路介面和安全群組。

如需搭配啟動範本使用標籤型政策的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[使用 IAM 許可控制啟動範本的存取](#)。

需要啟動範本和版本編號

您也可以使用 IAM 許可來強制規範在建立或更新 Auto Scaling 群組時，必須指定啟動範本和啟動範本的版本編號。

以下範例允許使用者只有在指定了啟動範本和啟動範本的版本編號時，才能建立和更新 Auto Scaling 群組。如果使用此政策的使用者省略版本號碼來指定 `$Latest` 或 `$Default` 啟動範本版本，或者改為嘗試使用啟動組態，則該動作會失敗。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```

        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": { "autoscaling:LaunchTemplateVersionSpecified": "true" }
    }
},
{
    "Effect": "Deny",
    "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup"
    ],
    "Resource": "*",
    "Condition": {
        "Null": { "autoscaling:LaunchConfigurationName": "false" }
    }
}
]
}

```

需要使用執行個體中繼資料服務版本第 2 版 (IMDSv2)

為了提高安全性，您可以設定使用者的權限，以要求使用需要 IMDSv2 的啟動範本。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[設定執行個體中繼資料服務](#)。

以下範例指出，除非也選擇加入執行個體以要求使用 IMDSv2 (由 "ec2:MetadataHttpTokens":"required" 指示)，否則使用者無法呼叫 ec2:RunInstances 動作。

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireImdsV2",
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:*:instance/*",

```



```
        "Condition": {
            "StringNotEquals": { "ec2:MetadataHttpTokens": "required" }
        }
    ]
}
```

 Tip

若要強制取代要啟動的 Auto Scaling 執行個體，其會使用新啟動範本或新版啟動範本 (與設定的執行個體中繼資料選項搭配使用)，則您可以啟動執行個體重新整理。如需詳細資訊，請參閱[更新 Auto Scaling 執行個體](#)。

限制對 Amazon EC2 資源的存取

下列範例示範如何使用資源層級許可和已標記的資源來限制對 Amazon EC2 資源的存取。

範例 1：將 Amazon EC2 執行個體啟動限制為特定資源和執行個體類型

以下範例會透過限制對 Amazon EC2 資源的存取，來控制使用者可啟動的執行個體組態。若要為啟動範本中指定的資源指定資源層級許可，您必須在 `RunInstances` 動作陳述式中包含這些資源。

在此例中，共有二個陳述式：

- 第一個陳述式要求使用者使用特定的安全群組 (**sg-903004f88example**)，以及使用特定的 AMI (**ami-04d5cc9b88example**)，將執行個體啟動至特定子網路 (**subnet-1a2b3c4d**)。它還可讓使用者存取啟動範本中指定的資源：網路介面、金鑰對和磁碟區。
- 第二個陳述式只允許使用者使用 **t2.micro** 或 **t2.small** 執行個體類型啟動執行個體，您可能會為了控制成本而這樣做。

不過，請注意，目前無法完全防止具有啟動範本啟動執行個體許可的使用者啟動其他執行個體類型。這是因為可以覆寫啟動範本中指定的執行個體類型，以使用使用屬性型執行個體類型選取項目定義的執行個體類型。

如需可以用來控制使用者可以啟動執行個體組態的資源層級許可完整清單，請參閱服務授權參考中的[適用於 Amazon EC2 的動作、資源及條件金鑰](#)。

範例 2：需要標籤並限制 Amazon EC2 執行個體啟動的資源存取

下列範例政策顯示如何使用身分型政策中的條件，根據標籤控制對 Amazon EC2 資源的存取。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "InstanceTags",
      "Effect": "Allow",
      "Action": [
        "ec2:RunInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/owner": "dev"
        }
      }
    },
    {
      "Sid": "InstanceBoundaries",
      "Effect": "Allow",
      "Action": [
        "ec2:RunInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1::image/*",
        "arn:aws:ec2:us-east-1:555555555555:subnet/*",
        "arn:aws:ec2:us-east-1:555555555555:network-interface/*"
      ]
    },
    {
      "Sid": "InstanceResourceTags",
      "Effect": "Allow",
      "Action": [
        "ec2:RunInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:key-pair/*",
        "arn:aws:ec2:us-east-1:555555555555:security-group/*",
```

```

        "arn:aws:ec2:us-east-1:555555555555:launch-template/*",
        "arn:aws:ec2:us-east-1:555555555555:volume/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/purpose": "testing"
        }
    }
}
]
}

```

在此範例中，有三個陳述式：

- 第一個陳述式只允許使用者在請求owner=dev中使用具有 的標籤時，在指定的區域中啟動執行個體。
- 第二個陳述式可讓使用者存取指定區域中的 AMI、子網路和網路界面。
- 第三個陳述式允許使用者在指定區域中使用具有標籤 的現有金鑰對、安全群組、啟動範本和磁碟區來啟動執行個體purpose=testing。

如需詳細資訊，請參閱《IAM 使用者指南》中的[使用標籤控制對 AWS 資源的存取](#)。

標記執行個體和磁碟區所需的許可

以下範例允許使用者在建立時標記執行個體和磁碟區。如果已在啟動範本中指定標籤，則會需要此政策。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[在建立過程中向標籤資源授予許可](#)。

其他啟動範本許可

您必須授予主控台使用者 ec2:DescribeLaunchTemplates 和 ec2:DescribeLaunchTemplateVersions 動作的許可。如果沒有這些許可，啟動範本資料就無法在 Auto Scaling 群組精靈中載入，且使用者也使用無法逐步執行精靈以使用啟動範本啟動執行個體。您可在 IAM 政策陳述式的 Action 元素中指定這些其他動作。

ec2:RunInstances 和 iam:PassRole 的許可驗證

使用者可以指定 Auto Scaling 群組使用的啟動範本版本。視其許可而定，此版本可以是特定編號的版本，或啟動範本的 \$Latest 或 \$Default 版本。如果是後者，請特別注意。這可能會覆寫您想要限制的 ec2:RunInstances 和 iam:PassRole 許可。

本章節將說明啟動範本的最新或預設版本與 Auto Scaling 群組搭配使用的案例。

當使用者呼叫 `CreateAutoScalingGroup`、`UpdateAutoScalingGroup` 或 `StartInstanceRefresh` API 時，Amazon EC2 Auto Scaling 會根據當時最新或預設版本的啟動範本版本檢查其許可，然後再繼續執行請求。這會驗證啟動執行個體時要完成之諸如 `ec2:RunInstances` 和 `iam:PassRole` 等動作的許可。為此我們發出 Amazon EC2 [RunInstances](#) 試轉呼叫，以驗證使用者是否具備進行動作所需的許可，而無需實際發出請求。傳回的回應由 Amazon EC2 Auto Scaling 讀取。如果使用者的許可不允許指定的動作，則 Amazon EC2 Auto Scaling 的請求會失敗，並傳回錯誤給使用者，其中包含缺少許可的相關資訊。

初始驗證和請求完成後，每當執行個體啟動時，Amazon EC2 Auto Scaling 都會使用其[服務連結角色](#)的許可，以最新或預設版本啟動執行個體 (即使已變更)。這表示使用啟動範本的使用者可能會更新該範本，以將 IAM 角色傳遞給執行個體 (即使執行個體沒有 `iam:PassRole` 許可)。

如果您要限制將群組設定為使用 `$Latest` 或 `$Default` 版本的使用者的存取權限，請使用 `autoscaling:LaunchTemplateVersionSpecified` 條件金鑰。這可確保 Auto Scaling 群組僅在使用者呼叫 `CreateAutoScalingGroup` 和 `UpdateAutoScalingGroup` API 時接受特定編號的版本。如需將此條件金鑰新增至 IAM 政策的範例，請參閱 [需要啟動範本和版本編號](#)。

對於設定為使用 `$Latest` 或 `$Default` 啟動範本版本的 Auto Scaling 群組，請考慮限制可建立和管理啟動範本版本的使用者，包括允許使用者指定預設啟動範本版本的 `ec2:ModifyLaunchTemplate` 動作。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[控制版本控制許可](#)。

相關資源

若要進一步了解檢視、建立和刪除啟動範本和啟動範本版本的許可，請參閱《Amazon EC2 使用者指南》中的[使用 IAM 許可控制啟動範本的存取](#)。

如需有關可用來控制 `RunInstances` 呼叫存取權的資源層級許可的詳細資訊，請參閱服務授權參考中的[適用於 Amazon EC2 的動作、資源及條件金鑰](#)。

在 Amazon EC2 執行個體上執行的應用程式的 IAM 角色

在 Amazon EC2 執行個體上執行的應用程式需要憑證才能存取其他 AWS 服務。若要以安全方式提供這些憑證，請使用 IAM 角色。該角色提供應用程式在存取其他 AWS 資源時可以使用的臨時許可。此角色的許可決定允許應用程式執行什麼。

對於 Auto Scaling 群組中的執行個體，您必須建立啟動範本或啟動組態，並選擇要與執行個體關聯的執行個體設定檔。執行個體設定檔是 IAM 角色的容器，而此角色允許 Amazon EC2 在執行個體啟動時

將 IAM 角色傳遞至該執行個體。首先，建立具有存取 AWS 資源所需所有許可的 IAM 角色。然後，建立執行個體設定檔並指派角色給此描述檔。

Note

作為最佳實務，我們強烈建議您建立角色，使其具有 AWS 服務應用程式所需的其他的最低許可。

目錄

- [先決條件](#)
- [建立啟動範本](#)
- [另請參閱](#)

先決條件

建立在 Amazon EC2 上執行的應用程式可以擔任的 IAM 角色。選擇適當的許可，以便後續獲得角色的應用程式可以進行其所需的特定 API 呼叫。

如果您使用 IAM 主控台而非 AWS CLI 或其中一個 AWS SDKs，主控台會自動建立執行個體描述檔，並為其提供與其對應之角色相同的名稱。

建立 IAM 角色 (主控台)

1. 前往網址 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 選擇建立角色。
4. 對於 Select trusted entity (選取信任的實體) 區段，選擇 AWS service (AWS 服務)。
5. 對於您的使用案例，選擇 EC2，然後選擇 Next (下一步)。
6. 可以的話，請選取用於許可政策的政策，或者選擇 Create policy (建立政策) 以開啟新的瀏覽器標籤，並從頭建立新的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立 IAM 政策](#)。在您建立政策後，關閉該標籤並返回您的原始標籤。選取您要服務具有之許可政策旁的核取方塊。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 實體許可界限](#)。
8. 選擇下一步。

9. 在 Name, review, and create (命名、檢閱和建立) 頁面的 Role name (角色名稱)，輸入角色名稱，以協助您識別此角色的用途。此名稱在您的 AWS 帳戶中必須是唯一的。由於其他 AWS 資源可能會參考角色，因此您無法在建立角色之後編輯角色的名稱。
10. 檢閱角色，然後選擇 Create role (建立角色)。

IAM 許可

使用 IAM 身分型政策來控制對新 IAM 角色的存取權。如果使用者是使用指定執行個體設定檔的啟動範本來建立或更新 Auto Scaling 群組的 IAM 身分 (使用者或角色)，則需要 `iam:PassRole` 許可。

下列政策範例會授與僅傳遞名稱開頭為 **gateam-** 之 IAM 角色的許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::111122223333:role/gateam-*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "ec2.amazonaws.com",
            "ec2.amazonaws.com.cn"
          ]
        }
      }
    }
  ]
}
```

Important

如需有關 Amazon EC2 Auto Scaling 如何驗證使用啟動範本之 Auto Scaling 群組的 `iam:PassRole` 動作許可的詳細資訊，請參閱 [ec2:RunInstances](#) 和 [iam:PassRole](#) 的許可驗證。

建立啟動範本

當您使用 建立啟動範本時 AWS 管理主控台，請在進階詳細資訊區段中，從 IAM 執行個體設定檔中選取角色。如需詳細資訊，請參閱[使用進階設定建立啟動範本](#)。

當您從 使用 [create-launch-template](#) 命令建立啟動範本時 AWS CLI，請指定 IAM 角色的執行個體描述檔名稱，如下列範例所示。

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

另請參閱

如需協助您開始學習和使用 Amazon EC2 IAM 角色的詳細資訊，請參閱：

- [《Amazon EC2 使用者指南》中的 Amazon EC2 的 IAM 角色](#) Amazon EC2
- [《IAM 使用者指南》中的使用執行個體設定檔](#)以及[使用 IAM 角色向在 Amazon EC2 執行個體上執行的應用程式授予許可](#)

Amazon EC2 Auto Scaling 的合規驗證

若要了解 是否 AWS 服務 在特定合規計劃的範圍內，請參閱[AWS 服務 合規計劃範圍內](#)然後選擇您感興趣的合規計劃。如需一般資訊，請參閱[AWS 合規計劃](#)。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱[下載報告 in AWS Artifact](#)

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用 時合規責任的詳細資訊 AWS 服務，請參閱 [AWS 安全文件](#)。

PCI DSS 合規

Amazon EC2 Auto Scaling 支援處理、儲存、傳輸商家或服務供應商的信用卡資料，並且已驗證符合支付卡產業 (PCI) 資料安全標準 (DSS)。如需 PCI DSS 的詳細資訊，包括如何請求 AWS PCI 合規套件的副本，請參閱 [PCI DSS 第 1 級](#)。

如需有關為 AWS 工作負載實現 PCI DSS 合規的資訊，請參閱下列合規指南：

- [上的支付卡產業資料安全標準 \(PCI DSS\) 3.2.1 AWS](#)

Amazon EC2 Auto Scaling 和介面 VPC 端點

您可以將 Amazon EC2 Auto Scaling 設定為使用介面 VPC 端點，進而提升 VPC 的安全狀態。介面端點採用 AWS PrivateLink 這項技術，可讓您將 VPC 和 Amazon EC2 Auto Scaling 之間的所有網路流量限制為 AWS 網路，以私密方式存取 Amazon EC2 Auto Scaling APIs。使用介面端點，您也不需要國際網路閘道、NAT 裝置或虛擬私有閘道。

您不需要設定 AWS PrivateLink，但建議使用。如需 AWS PrivateLink 和 VPC 端點的詳細資訊，請參閱《AWS PrivateLink 指南》中的[什麼是 AWS PrivateLink？](#)。

主題

- [建立介面 VPC 端點](#)
- [建立 VPC 端點政策](#)

建立介面 VPC 端點

使用下列服務名稱為 Amazon EC2 Auto Scaling 建立端點：

```
com.amazonaws.region.autoscaling
```

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的[使用介面 VPC 端點存取 AWS 服務](#)。

您不需要變更任何 Amazon EC2 Auto Scaling 設定。Amazon EC2 Auto Scaling 會使用 AWS 服務端點或私有介面 VPC 端點呼叫其他服務，以使用中者為準。

建立 VPC 端點政策

您可以將政策連接到 VPC 端點，以控制對 Amazon EC2 Auto Scaling API 的存取。此政策指定：

- 可執行動作的委託人。
- 可執行的動作。
- 可供執行動作的資源。

下列範例顯示 VPC 端點政策，拒絕所有人透過端點刪除擴展政策的許可。範例政策也會授予所有人執行所有其他動作的許可。


```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "autoscaling:DeleteScalingPolicy",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": "*"
    }
  ]
}
```

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的[使用端點政策控制 VPC 端點的存取](#)。

搭配 AWS SDK 使用此服務

AWS 軟體開發套件 (SDKs) 適用於許多熱門的程式設計語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
適用於 C++ 的 AWS SDK	適用於 C++ 的 AWS SDK 程式碼範例
AWS CLI	AWS CLI 程式碼範例
適用於 Go 的 AWS SDK	適用於 Go 的 AWS SDK 程式碼範例
適用於 Java 的 AWS SDK	適用於 Java 的 AWS SDK 程式碼範例
適用於 JavaScript 的 AWS SDK	適用於 JavaScript 的 AWS SDK 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
適用於 .NET 的 AWS SDK	適用於 .NET 的 AWS SDK 程式碼範例
適用於 PHP 的 AWS SDK	適用於 PHP 的 AWS SDK 程式碼範例
AWS Tools for PowerShell	AWS Tools for PowerShell 程式碼範例
適用於 Python (Boto3) 的 AWS SDK	適用於 Python (Boto3) 的 AWS SDK 程式碼範例
適用於 Ruby 的 AWS SDK	適用於 Ruby 的 AWS SDK 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

如需此服務的特定範例，請參閱 [Auto Scaling AWS SDKs 程式碼範例](#)。

 可用性範例

找不到所需的內容嗎？請使用本頁面底部的提供意見回饋連結申請程式碼範例。

Auto Scaling AWS SDKs程式碼範例

下列程式碼範例示範如何使用 Auto Scaling 搭配 AWS 軟體開發套件 (SDK)。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

程式碼範例

- [Auto Scaling AWS SDKs的基本範例](#)
 - [Hello Auto Scaling](#)
 - [了解使用 AWS SDK Auto Scaling 的基本概念](#)
 - [使用 AWS SDKs的 Auto Scaling 動作](#)
 - [搭配使用 AttachInstances 與 CLI](#)
 - [AttachLoadBalancerTargetGroups 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 AttachLoadBalancers 與 CLI](#)
 - [搭配使用 CompleteLifecycleAction 與 CLI](#)
 - [CreateAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 CreateLaunchConfiguration 與 CLI](#)
 - [搭配使用 CreateOrUpdateTags 與 CLI](#)
 - [DeleteAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 DeleteLaunchConfiguration 與 CLI](#)
 - [搭配使用 DeleteLifecycleHook 與 CLI](#)
 - [搭配使用 DeleteNotificationConfiguration 與 CLI](#)
 - [搭配使用 DeletePolicy 與 CLI](#)
 - [搭配使用 DeleteScheduledAction 與 CLI](#)
 - [搭配使用 DeleteTags 與 CLI](#)
 - [搭配使用 DescribeAccountLimits 與 CLI](#)

- [搭配使用 DescribeAdjustmentTypes 與 CLI](#)
- [DescribeAutoScalingGroups 搭配 AWS SDK 或 CLI 使用](#)
- [DescribeAutoScalingInstances 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 DescribeAutoScalingNotificationTypes 與 CLI](#)
- [搭配使用 DescribeLaunchConfigurations 與 CLI](#)
- [搭配使用 DescribeLifecycleHookTypes 與 CLI](#)
- [搭配使用 DescribeLifecycleHooks 與 CLI](#)
- [搭配使用 DescribeLoadBalancers 與 CLI](#)
- [搭配使用 DescribeMetricCollectionTypes 與 CLI](#)
- [搭配使用 DescribeNotificationConfigurations 與 CLI](#)
- [搭配使用 DescribePolicies 與 CLI](#)
- [DescribeScalingActivities 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 DescribeScalingProcessTypes 與 CLI](#)
- [搭配使用 DescribeScheduledActions 與 CLI](#)
- [搭配使用 DescribeTags 與 CLI](#)
- [搭配使用 DescribeTerminationPolicyTypes 與 CLI](#)
- [搭配使用 DetachInstances 與 CLI](#)
- [搭配使用 DetachLoadBalancers 與 CLI](#)
- [DisableMetricsCollection 搭配 AWS SDK 或 CLI 使用](#)
- [EnableMetricsCollection 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 EnterStandby 與 CLI](#)
- [搭配使用 ExecutePolicy 與 CLI](#)
- [搭配使用 ExitStandby 與 CLI](#)
- [搭配使用 PutLifecycleHook 與 CLI](#)
- [搭配使用 PutNotificationConfiguration 與 CLI](#)
- [搭配使用 PutScalingPolicy 與 CLI](#)
- [搭配使用 PutScheduledUpdateGroupAction 與 CLI](#)
- [搭配使用 RecordLifecycleActionHeartbeat 與 CLI](#)
- [搭配使用 ResumeProcesses 與 CLI](#)
- [SetDesiredCapacity 搭配 AWS SDK 或 CLI 使用](#)

- [搭配使用 SetInstanceHealth 與 CLI](#)
- [搭配使用 SetInstanceProtection 與 CLI](#)
- [搭配使用 SuspendProcesses 與 CLI](#)
- [TerminateInstanceInAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
- [UpdateAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
- [使用 AWS SDKs 的 Auto Scaling 案例](#)
 - [使用 AWS SDK 建置和管理彈性服務](#)

Auto Scaling AWS SDKs 的基本範例

下列程式碼範例示範如何搭配使用 Amazon EC2 Auto Scaling 的基本功能與 AWS SDK。

範例

- [Hello Auto Scaling](#)
- [了解使用 AWS SDK Auto Scaling 的基本概念](#)
- [使用 AWS SDKs 的 Auto Scaling 動作](#)
 - [搭配使用 AttachInstances 與 CLI](#)
 - [AttachLoadBalancerTargetGroups 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 AttachLoadBalancers 與 CLI](#)
 - [搭配使用 CompleteLifecycleAction 與 CLI](#)
 - [CreateAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 CreateLaunchConfiguration 與 CLI](#)
 - [搭配使用 CreateOrUpdateTags 與 CLI](#)
 - [DeleteAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 DeleteLaunchConfiguration 與 CLI](#)
 - [搭配使用 DeleteLifecycleHook 與 CLI](#)
 - [搭配使用 DeleteNotificationConfiguration 與 CLI](#)
 - [搭配使用 DeletePolicy 與 CLI](#)
 - [搭配使用 DeleteScheduledAction 與 CLI](#)
 - [搭配使用 DeleteTags 與 CLI](#)
 - [搭配使用 DescribeAccountLimits 與 CLI](#)

- [搭配使用 DescribeAdjustmentTypes 與 CLI](#)
- [DescribeAutoScalingGroups 搭配 AWS SDK 或 CLI 使用](#)
- [DescribeAutoScalingInstances 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 DescribeAutoScalingNotificationTypes 與 CLI](#)
- [搭配使用 DescribeLaunchConfigurations 與 CLI](#)
- [搭配使用 DescribeLifecycleHookTypes 與 CLI](#)
- [搭配使用 DescribeLifecycleHooks 與 CLI](#)
- [搭配使用 DescribeLoadBalancers 與 CLI](#)
- [搭配使用 DescribeMetricCollectionTypes 與 CLI](#)
- [搭配使用 DescribeNotificationConfigurations 與 CLI](#)
- [搭配使用 DescribePolicies 與 CLI](#)
- [DescribeScalingActivities 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 DescribeScalingProcessTypes 與 CLI](#)
- [搭配使用 DescribeScheduledActions 與 CLI](#)
- [搭配使用 DescribeTags 與 CLI](#)
- [搭配使用 DescribeTerminationPolicyTypes 與 CLI](#)
- [搭配使用 DetachInstances 與 CLI](#)
- [搭配使用 DetachLoadBalancers 與 CLI](#)
- [DisableMetricsCollection 搭配 AWS SDK 或 CLI 使用](#)
- [EnableMetricsCollection 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 EnterStandby 與 CLI](#)
- [搭配使用 ExecutePolicy 與 CLI](#)
- [搭配使用 ExitStandby 與 CLI](#)
- [搭配使用 PutLifecycleHook 與 CLI](#)
- [搭配使用 PutNotificationConfiguration 與 CLI](#)
- [搭配使用 PutScalingPolicy 與 CLI](#)
- [搭配使用 PutScheduledUpdateGroupAction 與 CLI](#)
- [搭配使用 RecordLifecycleActionHeartbeat 與 CLI](#)
- [搭配使用 ResumeProcesses 與 CLI](#)
- [SetDesiredCapacity 搭配 AWS SDK 或 CLI 使用](#)

- [搭配使用 SetInstanceHealth 與 CLI](#)
- [搭配使用 SetInstanceProtection 與 CLI](#)
- [搭配使用 SuspendProcesses 與 CLI](#)
- [TerminateInstanceInAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
- [UpdateAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)

Hello Auto Scaling

下列程式碼範例示範如何開始使用 Auto Scaling。

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling groups.");
    }
}
```



```
var response = await client.DescribeAutoScalingGroupsAsync();

if (response.AutoScalingGroups == null ||
response.AutoScalingGroups.Count == 0)
{
    Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
groups.");
    return;
}
response.AutoScalingGroups.ForEach(autoScalingGroup =>
{
    Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.AvailabilityZone}");
})
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

CMakeLists.txt CMake 檔案的程式碼。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)
```

```

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
    libraries for the AWS SDK.
        string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
            "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
        list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
    endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
    for running and debugging.

        # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
    may need to uncomment this

                                # and set the proper subdirectory to the
    executables' location.

        AWSSDK_COPY_DYN_LIBS(SERVICE_COMPONENTS ""
            ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
    endif ()

add_executable(${PROJECT_NAME}
    hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello_autoscaling.cpp 來源檔案的程式碼。

```
#include <aws/core/Aws.h>
```

```

#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 * A "Hello Autoscaling" starter application which initializes an Amazon EC2
 * Auto Scaling client and describes the
 * Amazon EC2 Auto Scaling groups.
 *
 * main function
 *
 * Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
                &autoScalingGroups =

```

```

        outcome.GetResult().GetAutoScalingGroups();
        for (auto &group: autoScalingGroups) {
            groupNames.push_back(group.GetAutoScalingGroupName());
        }
        nextToken = outcome.GetResult().GetNextToken();
    } else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups."
"
                << outcome.GetError().GetMessage()
                << std::endl;
        result = 1;
        break;
    }
} while (!nextToken.empty());

std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
std::endl;
for (auto &groupName: groupNames) {
    std::cout << "AutoScaling group: " << groupName << std::endl;
}

}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
        });
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DescribeAutoScalingGroups](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import boto3

def hello_autoscaling(autoscaling_client):
```

```
"""
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
    client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
        groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
            {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")

if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [DescribeAutoScalingGroups](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
# operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:                #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:          #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```


- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name: {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn: {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [DescribeAutoScalingGroups](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

了解使用 AWS SDK Auto Scaling 的基本概念

下列程式碼範例示範如何：

- 以啟動範本和可用區域建立 Amazon EC2 Auto Scaling 群組，並取得有關執行中執行個體的相關資訊。
- 啟用 Amazon CloudWatch 指標收集。
- 更新群組所需的容量，並等待執行個體啟動。
- 終止群組中的執行個體。
- 列出為因應使用者請求和容量變更而發生的擴展活動。
- 取得 CloudWatch 指標的統計資料，然後清除資源。

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;

using Amazon.EC2;
using Microsoft.Extensions.Configuration;
```

```
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
                        LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
                        LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonCloudWatch>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalingWrapper>()
                    .AddTransient<CloudWatchWrapper>()
                    .AddTransient<EC2Wrapper>()
                    .AddTransient<UIWrapper>()
            )
            .Build();

        var autoScalingWrapper =
            host.Services.GetRequiredService<AutoScalingWrapper>();
        var cloudWatchWrapper =
            host.Services.GetRequiredService<CloudWatchWrapper>();
        var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();
    }
}
```

```
var imageId = configuration["ImageId"];
var instanceType = configuration["InstanceType"];
var launchTemplateName = configuration["LaunchTemplateName"];

launchTemplateName += Guid.NewGuid().ToString();

// The name of the Auto Scaling group.
var groupName = configuration["GroupName"];

uiWrapper.DisplayTitle("Auto Scaling Basics");
uiWrapper.DisplayAutoScalingBasicsDescription();

// Create the launch template and save the template Id to use when
deleting the
// launch template at the end of the application.
var launchTemplateId = await
ec2Wrapper.CreateLaunchTemplateAsync(imageId!, instanceType!,
launchTemplateName);

// Confirm that the template was created by asking for a description of
it.
await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

uiWrapper.PressEnter();

var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
await autoScalingWrapper.CreateAutoScalingGroupAsync(
    groupName!,
    launchTemplateName,
    availabilityZones[0].ZoneName);

// Keep checking the details of the new group until its lifecycle state
// is "InService".
Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

List<AutoScalingInstanceDetails> instanceDetails;

do
{
    instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
}
```

```
while (instanceDetails.Count <= 0);

Console.WriteLine($"Auto scaling group {groupName} successfully
created.");
Console.WriteLine($"{instanceDetails.Count} instances were created for
the group.");

// Display the details of the Auto Scaling group.
instanceDetails.ForEach(detail =>
{
    Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
});

uiWrapper.PressEnter();

uiWrapper.DisplayTitle("Metrics collection");
Console.WriteLine($"Enable metrics collection for {groupName}");
await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);

// Show the metrics that are collected for the group.

// Update the maximum size of the group to three instances.
Console.WriteLine("--- Update the Auto Scaling group to increase max size
to 3 ---");
int maxSize = 3;
await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

Console.WriteLine("--- Describe all Auto Scaling groups to show the
current state of the group ---");
var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

uiWrapper.DisplayGroupDetails(groups!);

uiWrapper.PressEnter();

uiWrapper.DisplayTitle("Describe account limits");
await autoScalingWrapper.DescribeAccountLimitsAsync();

uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");

uiWrapper.DisplayTitle("Set desired capacity");
int desiredCapacity = 2;
```

```
        await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

        Console.WriteLine("Get the two instance Id values");

        // Empty the group before getting the details again.
        groups.Clear();
        groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
        if (groups.Any())
        {
            foreach (AutoScalingGroup group in groups)
            {
                Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
                Console.WriteLine($"The group ARN is
{group.AutoScalingGroupARN}");
                var instances = group.Instances;
                foreach (Amazon.AutoScaling.Model.Instance instance in instances)
                {
                    Console.WriteLine($"The instance id is
{instance.InstanceId}");
                    Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
                }
            }
        }

        uiWrapper.DisplayTitle("Scaling Activities");
        Console.WriteLine("Let's list the scaling activities that have occurred
for the group.");
        var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
        if (activities.Any())
        {
            activities.ForEach(activity =>
            {
                Console.WriteLine($"The activity Id is {activity.ActivityId}");
                Console.WriteLine($"The activity details are
{activity.Details}");
            });
        }

        // Display the Amazon CloudWatch metrics that have been collected.
```

```
        var metrics = await
cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
        if (metrics.Any())
        {
            Console.WriteLine($"Metrics collected for {groupName}:");
            metrics.ForEach(metric =>
            {
                Console.Write($"Metric name: {metric.MetricName}\t");
                Console.WriteLine($"Namespace: {metric.Namespace}");
            });
        }

        var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
        if (dataPoints.Any())
        {
            Console.WriteLine("Details for the metrics collected:");
            dataPoints.ForEach(detail => { Console.WriteLine(detail); });
        }

        // Disable metrics collection.
        Console.WriteLine("Disabling the collection of metrics for
{groupName}.");
        var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

        if (success)
        {
            Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
        }

        // Terminate all instances in the group.
        uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
        Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

        if (groups is not null)
        {
```

```

        groups.ForEach(group =>
        {
            // Only delete instances in the AutoScaling group we created.
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(async instance =>
                {
                    await
autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                });
            }
        });

        // After all instances are terminated, delete the group.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the Auto Scaling group.");
        await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);

        // Delete the launch template.
        var deletedLaunchTemplateName = await
ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }

        Console.WriteLine("The demo is now concluded.");
    }
}

namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>

```



```

    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
    {
        Console.WriteLine("This code example performs the following
operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
        Console.WriteLine(" 7. Changes the desired capacity of the Auto
Scaling");
        Console.WriteLine("    group to use an additional instance.");
        Console.WriteLine(" 8. Shows that there are now instances in the
group.");
        Console.WriteLine(" 9. Lists the scaling activities that have occurred
for the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that
have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>

```

```

        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>

```

```

    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}

```

定義案例呼叫的函式，以管理啟動範本和指標。這些函式包裝了 Auto Scaling、Amazon EC2 和 CloudWatch 動作。

```

namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
    client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)

```

```
{
    _amazonAutoScaling = amazonAutoScaling;
}

/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
    {
        availabilityZone,
    };

    var request = new CreateAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        AvailabilityZones = zoneList,
        LaunchTemplate = templateSpecification,
        MaxSize = 6,
        MinSize = 1
    };
    try
    {
        var response = await
            _amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AlreadyExistsException)
```

```

        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }

    /// <summary>
    /// Retrieve information about Amazon EC2 Auto Scaling quotas to the
    /// active AWS account.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DescribeAccountLimitsAsync()
    {
        var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
            response.MaxNumberOfAutoScalingGroups);
        Console.WriteLine("The current number of Auto Scaling groups is " +
            response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
    param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
            _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
    }

```

```

        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }

    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(

```

```

        scalingGroupsRequest);
    if (response.AutoScalingInstances != null)
    {
        instanceDetails = response.AutoScalingInstances;
    }
}

return instanceDetails;
}

/// <summary>
/// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
public async Task<List<AutoScalingGroup>> DescribeAutoScalingGroupsAsync(
    string groupName)
{
    var groups = new List<AutoScalingGroup>();
    var groupList = new List<string>
    {
        groupName,
    };

    var request = new DescribeAutoScalingGroupsRequest
    {
        AutoScalingGroupNames = groupList,
    };

    var response = await
        _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
    if (response.AutoScalingGroups != null)
    {
        groups = response.AutoScalingGroups;
    }

    return groups;
}

/// <summary>

```

```
/// Delete an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteAutoScalingGroupAsync(
    string groupName)
{
    var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        ForceDelete = true,
    };

    var response = await
_amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You successfully deleted {groupName}");
        return true;
    }

    Console.WriteLine($"Couldn't delete {groupName}.");
    return false;
}

/// <summary>
/// Disable the collection of metric data for an Amazon EC2 Auto Scaling
/// group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public async Task<bool> DisableMetricsCollectionAsync(string groupName)
{
    var request = new DisableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
    };

    var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```



```
}

/// <summary>
/// Enable the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> EnableMetricsCollectionAsync(string groupName)
{
    var listMetrics = new List<string>
    {
        "GroupMaxSize",
    };

    var collectionRequest = new EnableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
        Metrics = listMetrics,
        Granularity = "1Minute",
    };

    var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Set the desired capacity of an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="desiredCapacity">The desired capacity for the Auto
/// Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> SetDesiredCapacityAsync(
    string groupName,
    int desiredCapacity)
{
    var capacityRequest = new SetDesiredCapacityRequest
    {
        AutoScalingGroupName = groupName,
        DesiredCapacity = desiredCapacity,
    };
};
```

```

        var response = await
        _amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
        {desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
        _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }

    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>

```

```
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
/// <param name="maxSize">The maximum number of instances that can be
/// created for the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> UpdateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    int maxSize)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var groupRequest = new UpdateAutoScalingGroupRequest
    {
        MaxSize = maxSize,
        AutoScalingGroupName = groupName,
        LaunchTemplate = templateSpecification,
    };

    var response = await
        _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
        return true;
    }
    else
    {
        return false;
    }
}

}

namespace AutoScalingActions;

using Amazon.EC2;
using Amazon.EC2.Model;
```

```
public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }

    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Delete an Amazon EC2 launch template.

```

```

    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
    {
        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string
launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.Write($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
    }

```

```
}

/// <summary>
/// Retrieve the availability zones for the current region.
/// </summary>
/// <returns>A collection of availability zones.</returns>
public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
{
    var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
        new DescribeAvailabilityZonesRequest());

    return response.AvailabilityZones;
}
}

namespace AutoScalingActions;

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</
returns>
```

```

public async Task<List<Metric>> GetCloudWatchMetricsAsync(string groupName)
{
    var metrics = new List<Metric>();
    var filter = new DimensionFilter
    {
        Name = "AutoScalingGroupName",
        Value = $"{groupName}",
    };

    var request = new ListMetricsRequest
    {
        MetricName = "AutoScalingGroupName",
        Dimensions = new List<DimensionFilter> { filter },
        Namespace = "AWS/AutoScaling",
    };

    var response = await _amazonCloudWatch.ListMetricsAsync(request);
    if (response.Metrics != null)
    {
        metrics = response.Metrics;
    }
    return metrics;
}

/// <summary>
/// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A list of data points.</returns>
public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
{
    var dataPoints = new List<Datapoint>();
    var metricDimensions = new List<Dimension>
    {
        new Dimension
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        },
    };

    // The start time will be yesterday.
    var startTime = DateTime.UtcNow.AddDays(-1);

```

```
var request = new GetMetricStatisticsRequest
{
    MetricName = "AutoScalingGroupName",
    Dimensions = metricDimensions,
    Namespace = "AWS/AutoScaling",
    Period = 60, // 60 seconds.
    Statistics = new List<string>() { "Minimum" },
    StartTimeUtc = startTime,
    EndTimeUtc = DateTime.UtcNow,
};

var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);
if (response.Datapoints != null)
{
    dataPoints = response.Datapoints;
}

return dataPoints;
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的下列主題。

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

C++

適用於 C++ 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
    \sa groupsAndInstancesScenario()
    \param clientConfig: AWS client configuration.
    \return bool: Successful completion.
*/
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;
    std::cout
        << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
Scaling "
        << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
        << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
        "Would you like to use an existing EC2 launch template (y/n)? ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
            "Enter the name of the existing EC2 launch template. ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateNames(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =

```

```

        ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                << "' exists by calling DescribeLaunchTemplate." <<
std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template.
"
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }
    else { // 2. Or create a new EC2 launch template.
        templateName = askQuestion("Enter the name for a new EC2 launch template:
");

        Aws::EC2::Model::CreateLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;
requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
        requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

        request.SetLaunchTemplateData(requestLaunchTemplateData);

        Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
            ec2Client.CreateLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << " was
created."
                << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
            "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already
exists"
                << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "

```

```

        << outcome.GetError().GetMessage()
        << std::endl;
    }
}
Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
std::cout << "Let's create an Auto Scaling group." << std::endl;
Aws::String groupName = askQuestion(
    "Enter a name for the Auto Scaling group: ");
// 3. Retrieve a list of EC2 Availability Zones.
Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
{
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
        ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout
            << "EC2 instances can be created in the following
Availability Zones:"
            << std::endl;

        availabilityZones = outcome.GetResult().GetAvailabilityZones();
        for (size_t i = 0; i < availabilityZones.size(); ++i) {
            std::cout << "    " << i + 1 << ". "
                << availabilityZones[i].GetZoneName() << std::endl;
        }
    }
    else {
        std::cerr << "Error with EC2::DescribeAvailabilityZones. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources("", templateName, autoScalingClient, ec2Client);
        return false;
    }
}

int availabilityZoneChoice = askQuestionForIntRange(
    "Choose an Availability Zone: ", 1,
    static_cast<int>(availabilityZones.size()));
// 4. Create an Auto Scaling group with the specified Availability Zone.
{
    Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);

```

```

        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
            availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
            autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
            Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
        autoScalingClient)) {
        std::cout << "Here is the Auto Scaling group description." << std::endl;
        if (!autoScalingGroups.empty()) {
            logAutoScalingGroupInfo(autoScalingGroups);
        }
    }
    else {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    }

```

```

        return false;
    }

    std::cout
        << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
        << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    bool enableMetrics = askYesNoQuestion(
        "Do you want to collect metrics about the A"
        "Auto Scaling group during this demo (y/n)? ");
    // 7. Optionally enable metrics collection for the Auto Scaling group.
    if (enableMetrics) {
        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
            autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }
}

```

```

    std::cout << "Let's update the maximum number of EC2 instances in '" <<
    groupName <<
        "' from 1 to 3." << std::endl;
    askQuestion("Press enter to continue: ", alwaysTrueTest);
    // 8. Update the Auto Scaling group, setting a new maximum size.
    {
        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
            autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
        autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            const auto &instances = autoScalingGroups[0].GetInstances();
            std::cout
                << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                << "No EC2 launch groups were retrieved from DescribeGroup
request."
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }
}

```

```

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
    << std::endl;
std::cout << "Let's update the desired capacity in '" << groupName <<
    "' from 1 to 2." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 9. Update the Auto Scaling group, setting a new desired capacity.
{
    Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetDesiredCapacity(2);

    Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
        autoScalingClient.SetDesiredCapacity(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
    autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        std::cout
            << "Here is the current state of the group." << std::endl;
        logAutoScalingGroupInfo(autoScalingGroups);
    }
    else {
        std::cerr
            << "No EC2 launch groups were retrieved from DescribeGroup
request."
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

std::cout << "Waiting for the new EC2 instance to start..." << std::endl;

```

```

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
    "\n"
        << std::endl;

    std::cout << "Let's terminate one of the EC2 instances in " << groupName <<
    "."
        << std::endl;
    std::cout << "Because the desired capacity is 2, another EC2 instance will
start "
        << "to replace the terminated EC2 instance."
        << std::endl;
    std::cout << "The currently running EC2 instances are:" << std::endl;

    if (autoScalingGroups.empty()) {
        std::cerr << "Error describing groups. No groups returned." << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    int instanceNumber = 1;
    Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
        autoScalingGroups[0].GetInstances());
    for (const Aws::String &instanceID: instanceIDs) {
        std::cout << "    " << instanceNumber << ". " << instanceID << std::endl;
        ++instanceNumber;
    }

    instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to
stop? ",
                                           1,
                                           static_cast<int>(instanceIDs.size()));

    // 10. Terminate an EC2 instance in the Auto Scaling group.
    {
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =

```



```

        autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                << instanceIDs[instanceNumber - 1] << "' to terminate..."
                << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
        << std::endl;
    std::cout << "Let's get a report of scaling activities for EC2 launch group
'"
        << groupName << "'."
        << std::endl;
    askQuestion("Press enter to continue: ", alwaysTrueTest);
    // 11. Get a description of activities for the Auto Scaling group.
    {
        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {

```

```

        const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
            outcome.GetResult().GetActivities();
        allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
} while (!nextToken.empty());

std::cout << "Found " << allActivities.size() << " activities."
    << std::endl;
std::cout << "Activities are ordered with the most recent first."
    << std::endl;
for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
    std::cout << activity.GetDescription() << std::endl;
    std::cout << activity.GetDetails() << std::endl;
}
}

if (enableMetrics) {
    if (!logAutoScalingMetrics(groupName, clientConfig)) {
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
}

std::cout << "Let's clean up." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);

// 13. Disable metrics collection if enabled.
if (enableMetrics) {
    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

```

```

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
            autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    return cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
    \sa waitInstances()
    \param groupName: An Auto Scaling group name.
    \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
    \param client: 'AutoScalingClient' instance.
    \return bool: Successful completion.
*/
bool AwsDoc::AutoScaling::waitInstances(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                     const
    Aws::AutoScaling::AutoScalingClient &client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }
    }
}

```

```

    }

    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++count;
    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }
    Aws::Vector<Aws::String> instanceIDs;
    if (!autoScalingGroups.empty()) {
        instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
        desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
    }

    if (instanceIDs.empty()) {
        if (desiredCapacity == 0) {
            break;
        }
        else {
            if ((count % 5) == 0) {
                std::cout << "No instance IDs returned for group." <<
std::endl;
            }

            continue;
        }
    }
}

// 6. Check lifecycle state of the instances using
DescribeAutoScalingInstances.
Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
request.SetInstanceIds(instanceIDs);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
    client.DescribeAutoScalingInstances(request);

if (outcome.IsSuccess()) {
    const
    Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
    &instancesDetails =
        outcome.GetResult().GetAutoScalingInstances();
    ready = instancesDetails.size() >= desiredCapacity;
    for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails
    &details: instancesDetails) {

```

```

        if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
            ready = false;
            break;
        }
    }
    // Log the status while waiting.
    if (((count % 5) == 1) || ready) {
        logInstancesLifecycleState(instancesDetails);
    }
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                << outcome.GetError().GetMessage()
                << std::endl;
    return false;
}
}

if (!describeGroup(groupName, autoScalingGroups, client)) {
    return false;
}

return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
    \sa cleanupResources()
    \param groupName: Optional Auto Scaling group name.
    \param templateName: Optional EC2 launch template name.
    \param autoScalingClient: 'AutoScalingClient' instance.
    \param ec2Client: 'EC2Client' instance.
    \return bool: Successful completion.
    */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const
                                           Aws::AutoScaling::AutoScalingClient &autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client)
{
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&

```

```

        (askYesNoQuestion(
            Aws::String("Delete the Auto Scaling group '" + groupName +
                "' (y/n)?")))) {
    {
        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMinSize(0);
        request.SetDesiredCapacity(0);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
            autoScalingClient.UpdateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout
                << "The minimum size and desired capacity of the Auto
Scaling group "
                << "was set to zero before terminating the instances."
                << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
        autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                autoScalingGroups[0].GetInstances());
            for (const Aws::String &instanceID: instanceIDs) {

                Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                request.SetInstanceId(instanceID);
                request.SetShouldDecrementDesiredCapacity(true);

                Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =
                    autoScalingClient.TerminateInstanceInAutoScalingGroup(
                        request);
            }
        }
    }
}

```

```

        if (outcome.IsSuccess()) {
            std::cout << "Initiating termination of EC2 instance '"
                << instanceID << "'." << std::endl;
        }
        else {
            std::cerr
                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    std::cout
        << "Waiting for the EC2 instances to terminate before
deleting the "
        << "Auto Scaling group..." << std::endl;
    waitForInstances(groupName, autoScalingGroups, autoScalingClient);
}

{
    Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
        autoScalingClient.DeleteAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

}

// 15. Delete the EC2 launch template.
if (!templateName.empty() && (askYesNoQuestion(

```

```

        Aws::String("Delete the EC2 launch template '" + templateName +
            "' (y/n)?")) {
    Aws::EC2::Model::DeleteLaunchTemplateRequest request;
    request.SetLaunchTemplateName(templateName);

    Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
        ec2Client.DeleteLaunchTemplate(request);

    if (outcome.IsSuccess()) {
        std::cout << "EC2 launch template '" << templateName << "' was
deleted."
                << std::endl;
    }
    else {
        std::cerr << "Error with EC2::DeleteLaunchTemplate. "
                << outcome.GetError().GetMessage()
                << std::endl;
        result = false;
    }
}

return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
\sa describeGroup()
\param groupName: An Auto Scaling group name.
\param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
\param client: 'AutoScalingClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
                                   const Aws::AutoScaling::AutoScalingClient
&client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =

```



```
        client.DescribeAutoScalingGroups(request);

        if (outcome.IsSuccess()) {
            autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }

        return outcome.IsSuccess();
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的下列主題。

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");
    private static final String ROLES_STACK = "MyCdkAutoScaleStack";
    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

        Usage:
            <groupName>

        Where:
            groupName - The name of the Auto Scaling group.

```

```

        """;

String groupName = "MyAutoScalingGroup2";
AutoScalingClient autoScalingClient = AutoScalingClient.builder()
    .region(Region.US_WEST_2)
    .build();

Ec2Client ec2 = Ec2Client.builder()
    .region(Region.US_WEST_2)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("First, we will create a launch template using a
CloudFormation script");
CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
String launchTemplateName = stackOutputs.get("LaunchTemplateNameOutput");
String vpcZoneId = getVPC(ec2);
updateTemplate(ec2, launchTemplateName );
System.out.println("The VPC zone id created by the CloudFormation stack
is"+vpcZoneId);

System.out.println("1. Create an Auto Scaling group named " + groupName);
createAutoScalingGroup(autoScalingClient, ec2, groupName,
launchTemplateName, vpcZoneId);

System.out.println(
    "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
Thread.sleep(60000);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get Auto Scale group Id value");
String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
if (instanceId.compareTo("") == 0) {
    System.out.println("Error - no instance Id value");
}

```

```
        System.exit(1);
    } else {
        System.out.println("The instance Id value is " + instanceId);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
    describeAutoScalingInstance(autoScalingClient, instanceId);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Enable metrics collection " + instanceId);
    enableMetricsCollection(autoScalingClient, groupName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Update an Auto Scaling group to update max size to
3");
    updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Describe Auto Scaling groups");
    describeAutoScalingGroups(autoScalingClient, groupName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Describe account details");
    describeAccountLimits(autoScalingClient);
    System.out.println(
        "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
    Thread.sleep(60000);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("8. Set desired capacity to 2");
    setDesiredCapacity(autoScalingClient, groupName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("9. Get the two instance Id values and state");
```

```

        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred
for the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling
group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group and cloud formation
resources");
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static String getVPC(Ec2Client ec2) {
        try {
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .filters(f -> f.name("isDefault").values("true"))
                .build();

            DescribeVpcsResponse response = ec2.describeVpcs(request);

            if (!response.vpcs().isEmpty()) {
                Vpc defaultVpc = response.vpcs().get(0);
            }
        }
    }

```

```

        System.out.println("Default VPC ID: " + defaultVpc.vpcId());
        return defaultVpc.vpcId();
    } else {
        System.out.println("No default VPC found.");
        return null; // Return null if no default VPC is found
    }

    } catch (Ec2Exception e) {
        System.err.println("EC2 error: " +
e.awsErrorDetails().errorMessage());
        return null; // Return null in case of an error
    }
}

public static void updateTemplate(Ec2Client ec2, String launchTemplateName) {
    // Step 1: Create new launch template version
    String newAmiId = "ami-0025f0db847eb6254";
    RequestLaunchTemplateData launchTemplateData =
RequestLaunchTemplateData.builder()
        .imageId(newAmiId)
        .build();

    CreateLaunchTemplateVersionRequest createVersionRequest =
CreateLaunchTemplateVersionRequest.builder()
        .launchTemplateName(launchTemplateName)
        .versionDescription("Updated with valid AMI")
        .sourceVersion("1")
        .launchTemplateData(launchTemplateData)
        .build();

    CreateLaunchTemplateVersionResponse createResponse =
ec2.createLaunchTemplateVersion(createVersionRequest);
    int newVersionNumber =
createResponse.launchTemplateVersion().versionNumber().intValue();

    // Step 2: Modify default version
    ModifyLaunchTemplateRequest modifyRequest =
ModifyLaunchTemplateRequest.builder()
        .launchTemplateName(launchTemplateName)
        .defaultVersion(String.valueOf(newVersionNumber))
        .build();

    ec2.modifyLaunchTemplate(modifyRequest);
}

```

```
        System.out.println("Updated launch template to version " +
newVersionNumber + " with AMI " + newAmiId);
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " +
activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                .autoScalingGroupName(groupName)
                .desiredCapacity(2)
                .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```

        System.exit(1);
    }
}

public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,

                                         Ec2Client ec2Client,
                                         String groupName,
                                         String launchTemplateName,
                                         String vpcId) {
    try {
        // Step 1: Get one subnet ID in the given VPC
        DescribeSubnetsRequest subnetRequest =
        DescribeSubnetsRequest.builder()
            .filters(Filter.builder().name("vpc-
id").values(vpcId).build())
            .build();

        DescribeSubnetsResponse subnetResponse =
        ec2Client.describeSubnets(subnetRequest);

        if (subnetResponse.subnets().isEmpty()) {
            throw new RuntimeException("No subnets found in VPC: " + vpcId);
        }

        String subnetId = subnetResponse.subnets().get(0).subnetId(); // Use
first subnet
        System.out.println("Using subnet: " + subnetId);

        // Step 2: Create launch template reference
        LaunchTemplateSpecification templateSpecification =
        LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        // Step 3: Create Auto Scaling group
        CreateAutoScalingGroupRequest request =
        CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .minSize(1)
            .maxSize(1)
            .vpcZoneIdentifier(subnetId) // Correct: subnet ID, not VPC
ID
    }
}

```



```

        .build();

        autoScalingClient.createAutoScalingGroup(request);

        // Step 4: Wait until group is created
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
            waiter.waitUntilGroupExists(groupsRequest);

        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (Ec2Exception | AutoScalingException e) {
        System.err.println("Error: " + e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient

        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

```

```
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .maxRecords(10)
            .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("*** The service to use for the health checks:
" + group.healthCheckType());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response = autoScalingClient
            .describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " +
group.autoScalingGroupARN());
        }
    }
}
```

```
        List<Instance> instances = group.instances();

        for (Instance instance : instances) {
            instanceId = instance.instanceId();
            System.out.println("The instance id is " + instanceId);
            System.out.println("The lifecycle state is " +
instance.lifecycleState());
        }
    }

    return instanceId;
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
```

```

        .metrics("GroupMaxSize")
        .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
    System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAccountLimits(AutoScalingClient autoScalingClient)
{
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is " +
response.numberOfWorkingAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)

```

```

        .launchTemplate(templateSpecification)
        .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group
" + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName) {
    try {

```

```
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build();

autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-
linked role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances
in the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("***** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
    delay(60000)
}
```

```
val instanceId = getSpecificAutoScaling(groupName)
if (instanceId.compareTo("") == 0) {
    println("Error - no instance Id value")
    exitProcess(1)
} else {
    println("The instance Id value is $instanceId")
}

println("***** Describe Auto Scaling with the Id value $instanceId")
describeAutoScalingInstance(instanceId)

println("***** Enable metrics collection $instanceId")
enableMetricsCollection(groupName)

println("***** Update an Auto Scaling group to maximum size of 3")
updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

println("***** Describe all Auto Scaling groups to show the current state of
the groups")
describeAutoScalingGroups(groupName)

println("***** Describe account details")
describeAccountLimits()

println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
delay(60000)

println("***** Set desired capacity to 2")
setDesiredCapacity(groupName)

println("***** Get the two instance Id values and state")
getAutoScalingGroups(groupName)

println("***** List the scaling activities that have occurred for the group")
describeScalingActivities(groupName)

println("***** Terminate an instance in the Auto Scaling group")
terminateInstanceInAutoScalingGroup(instanceId)

println("***** Stop the metrics collection")
disableMetricsCollection(groupName)

println("***** Delete the Auto Scaling group")
```



```
        deleteSpecificAutoScalingGroup(groupName)
    }

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
```

```

        println("The activity Id is ${activity.activityId}")
        println("The activity details are ${activity.details}")
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {

```

```

    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
        }
}

```

```

        serviceLinkedRoleArn = serviceLinkedRoleARNVal
    }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

```

```

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
            ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
            ${response.numberOfWorkingAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

```

```
    }  
}  
  
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {  
    val deleteAutoScalingGroupRequest =  
        DeleteAutoScalingGroupRequest {  
            autoScalingGroupName = groupName  
            forceDelete = true  
        }  
  
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)  
        println("You successfully deleted $groupName")  
    }  
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("-----\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
PHP!\n");
        echo("-----\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**

```

```

    * Step 0: Create an EC2 launch template that you'll use to create an
    Auto Scaling group.
    */
    $this->ec2Client = new EC2Client($clientArgs);
    $this->templateName = "example_launch_template_$uniqid";
    $instanceType = "t1.micro";
    $amiId = "ami-0ca285d4c2cda3300";
    $launchTemplate = $this->ec2Client->createLaunchTemplate(
        [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
        ]
    );

    /**
     * Step 1: CreateAutoScalingGroup: pass it the launch template you
     created in step 0.
     */
    $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

    $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
    $minSize = 1;
    $maxSize = 1;
    $launchTemplateId = $launchTemplate['LaunchTemplate']
['LaunchTemplateId'];
    $this->autoScalingService->createAutoScalingGroup(
        $this->autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    );

    $this->autoScalingService->waitUntilGroupInService([$this->
autoScalingGroupName]);
    $autoScalingGroup = $this->autoScalingService->
describeAutoScalingGroups([$this->autoScalingGroupName]);

    /**

```



```
    * Step 2: DescribeAutoScalingInstances: show that one instance has
    launched.
    */
    $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
    $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
    echo "The Auto Scaling group {$this->autoScalingGroupName} was created
    successfully.\n";
    echo count($instances['AutoScalingInstances']) . " instances were created
    for the group.\n";
    echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
    number of instances for the group.\n";

    /**
    * Step 3: EnableMetricsCollection: enable all metrics or a subset.
    */
    $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

    /**
    * Step 4: UpdateAutoScalingGroup: update max size to 3.
    */
    echo "Updating the max number of instances to 3.\n";
    $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

    /**
    * Step 5: DescribeAutoScalingGroups: show the current state of the
    group.
    */
    $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
    echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
    echo " is the updated max number of instances for the group.\n";

    $limits = $this->autoScalingService->describeAccountLimits();
    echo "Here are your account limits:\n";
    echo "MaxNumberOfAutoScalingGroups:
{$limits['MaxNumberOfAutoScalingGroups']}\n";
    echo "MaxNumberOfLaunchConfigurations:
{$limits['MaxNumberOfLaunchConfigurations']}\n";
    echo "NumberOfAutoScalingGroups:
{$limits['NumberOfAutoScalingGroups']}\n";
```

```

        echo "NumberOfLaunchConfigurations:
        {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this-
>autoScalingGroupName, 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
         launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
            {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
{$instance['LifecycleState']}\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
         instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);

```

```

        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
    } while (count($instances) < 2);
    $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
    foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
        echo "There is a group named:
        {$autoScalingGroup['AutoScalingGroupName']}";
        echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\\n";
        foreach ($autoScalingGroup['Instances'] as $instance) {
            echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
with id of: ";
            echo "{$instance['InstanceId']} and a lifecycle state of:
{$instance['LifecycleState']}.\\n";
        }
    }

    /**
     * Step 9: DescribeScalingActivities: list the scaling activities that
     have occurred for the group so far.
     */
    $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
    echo "We found " . count($activities['Activities']) . " activities.\\n";
    foreach ($activities['Activities'] as $activity) {
        echo "{$activity['ActivityId']} - {$activity['StartTime']} -
{$activity['Description']}\\n";
    }

    /**
     * Step 10: Use the Amazon CloudWatch API to get and show some metrics
     collected for the group.
     */
    $metricsNamespace = 'AWS/AutoScaling';
    $metricsDimensions = [
        [
            'Name' => 'AutoScalingGroupName',
            'Value' => $autoScalingGroup['AutoScalingGroupName'],
        ],
    ];
    $metrics = $this->cloudWatchClient->listMetrics(
        [
            'Dimensions' => $metricsDimensions,

```

```

        'Namespace' => $metricsNamespace,
    ]
);
foreach ($metrics['Metrics'] as $metric) {
    $timespan = 5;
    if ($metric['MetricName'] != 'GroupTotalCapacity' &&
$metric['MetricName'] != 'GroupMaxSize') {
        continue;
    }
    echo "Over the last $timespan minutes, {$metric['MetricName']}
recorded:\n";
    $stats = $this->cloudWatchClient->getMetricStatistics(
        [
            'Dimensions' => $metricsDimensions,
            'EndTime' => time(),
            'StartTime' => time() - (5 * 60),
            'MetricName' => $metric['MetricName'],
            'Namespace' => $metricsNamespace,
            'Period' => 60,
            'Statistics' => ['Sum'],
        ]
    );
    foreach ($stats['Datapoints'] as $stat) {
        echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
    }
}

return $instances;
}

public function cleanUp()
{
    /**
     * Step 11: DisableMetricsCollection: disable all metrics.
     */
    $this->autoScalingService->disableMetricsCollection($this->autoScalingGroupName);

    /**
     * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
instances.
     * - UpdateAutoScalingGroup with MinSize=0
     * - TerminateInstanceInAutoScalingGroup for each instance,

```

```

        *      specify ShouldDecrementDesiredCapacity=True. Wait for instances to
stop.
        * - Now you can delete the group.
        */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
        * Step 13: Delete launch template.
        */
        $this->ec2Client->deleteLaunchTemplate(
            [
                'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}

```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的下列主題。
- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)

- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) ->
    None:
    """
        Runs the scenario demonstrating the management of Auto Scaling groups and
        instances.

        :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
        Scaling groups.
        :param svc_helper: An instance of the ServiceHelper that interacts with AWS
        services.
        :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")

    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
        instances."
```

```

    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
new one: "
    )
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following
specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
    print("-" * 88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask(
        "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) +
1)
    )
    group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_autoscaling_group(group_name, group_zones, template_name,
1, 1)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Created Auto Scaling group %s.", group_name)
    print("Created group:")

```

```
pp(group)
print("Waiting for instance to start...")
wait_for_group(group_name, as_wrapper)
print("-" * 88)

use_metrics = q.ask(
    "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
    q.is_yesno,
)
if use_metrics:
    as_wrapper.enable_metrics(
        group_name,
        [
            "GroupMinSize",
            "GroupMaxSize",
            "GroupDesiredCapacity",
            "GroupInServiceInstances",
            "GroupTotalInstances",
        ],
    )
    logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
    print(f"Metrics enabled for {group_name}.")
print("-" * 88)

print(f"Let's update the maximum number of instances in {group_name} from 1
to 3.")
q.ask("Press Enter when you're ready.")
as_wrapper.update_group(group_name, MaxSize=3)
group = as_wrapper.describe_group(group_name)
logger.info("Updated maximum size for group %s to 3.", group_name)
print("The group still has one running instance, but can have up to three:")
print_simplified_group(group)
print("-" * 88)

print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
q.ask("Press Enter when you're ready.")
as_wrapper.set_desired_capacity(group_name, 2)
wait(10)
group = as_wrapper.describe_group(group_name)
logger.info("Set desired capacity for group %s to 2.", group_name)
print("Here's the current state of the group:")
print_simplified_group(group)
print("-" * 88)
```



```

print("Waiting for the new instance to start...")
instance_ids = wait_for_group(group_name, as_wrapper)
print("-" * 88)

print(f"Let's terminate one of the instances in {group_name}.")
print("Because the desired capacity is 2, another instance will start.")
print("The currently running instances are:")
for index, inst_id in enumerate(instance_ids):
    print(f"\t{index+1}. {inst_id}")
inst_sel = q.ask(
    "Which instance do you want to stop? ",
    q.is_int,
    q.in_range(1, len(instance_ids) + 1),
)
print(f"Stopping {instance_ids[inst_sel-1]}...")
as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
wait(10)
group = as_wrapper.describe_group(group_name)
logger.info(
    "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
    group_name
)
print(f"Here's the state of {group_name}:")
print_simplified_group(group)
print("Waiting for the scaling activities to complete...")
wait_for_group(group_name, as_wrapper)
print("-" * 88)

print(f"Let's get a report of scaling activities for {group_name}.")
q.ask("Press Enter when you're ready.")
activities = as_wrapper.describe_scaling_activities(group_name)
logger.info(
    "Retrieved %d scaling activities for group %s.", len(activities),
    group_name
)
print(
    f"Found {len(activities)} activities.\n"
    f"Activities are ordered with the most recent one first:"
)
for act in activities:
    pp(act)
print("-" * 88)

if use_metrics:

```



```

as_wrapper.update_group(group_name, MinSize=0)
group = as_wrapper.describe_group(group_name)
instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
for inst_id in instance_ids:
    print(f"Stopping {inst_id}.")
    as_wrapper.terminate_instance(inst_id, True)
    logger.info("Terminated instance %s in group %s.", inst_id, group_name)
print("Waiting for instances to stop...")
wait_for_instances(instance_ids, as_wrapper)
print(f"Deleting {group_name}.")
as_wrapper.delete_autoscaling_group(group_name)
logger.info("Deleted Auto Scaling group %s.", group_name)
print("-" * 88)

if template is not None:
    if q.ask(
        f"Do you want to delete launch template {template_name} used in this
demo (y/n)? "
    ):
        svc_helper.delete_template(template_name)
        logger.info("Deleted launch template %s.", template_name)
        print("Template deleted.")

print("\nThanks for watching!")
print("-" * 88)

if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")

```

定義案例呼叫的函式，以管理啟動範本和指標。這些函式包裝了 Amazon EC2 和 CloudWatch 動作。

```

class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""

    def __init__(self, ec2_client, cloudwatch_resource):

```

```

    """
    :param ec2_client: A Boto3 Amazon EC2 client.
    :param cloudwatch_resource: A Boto3 CloudWatch resource.
    """
    self.ec2_client = ec2_client
    self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for
        instances
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateName=[template_name]
            )
            template = response["LaunchTemplates"][0]
            logger.info("Launch template %s retrieved successfully.",
template_name)
            return template
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                logger.warning("Launch template %s does not exist.",
template_name)
            else:
                logger.error(
                    "Couldn't verify launch template %s. Error: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
    """

```

Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.

```

:param template_name: The name to give to the template.
:param inst_type: The type of the instance, such as t1.micro.
:param ami_id: The ID of the Amazon Machine Image (AMI) to use when
creating
                an instance.
:return: Information about the newly created template.
:raises ClientError: If there is an error creating the launch template.
"""
try:
    response = self.ec2_client.create_launch_template(
        LaunchTemplateName=template_name,
        LaunchTemplateData={"InstanceType": inst_type, "ImageId":
ami_id},
    )
    template = response["LaunchTemplate"]
    logger.info(
        "Created launch template %s with instance type %s and AMI ID
%s.",
        template_name,
        inst_type,
        ami_id,
    )
    return template
except ClientError as err:
    logger.error(
        "Couldn't create launch template %s. Error: %s: %s",
        template_name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

def delete_template(self, template_name: str) -> None:
    """
    Deletes a launch template.

    :param template_name: The name of the template to delete.
    :raises ClientError: If there is an error deleting the launch template.
    """
    try:

```

```

self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
    logger.info("Deleted launch template %s.", template_name)
except ClientError as err:
    logger.error(
        "Couldn't delete launch template %s. Error: %s: %s",
        template_name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

def get_availability_zones(self) -> list:
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    :raises ClientError: If there is an error retrieving availability zones.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        logger.info("Retrieved availability zones: %s.", ", ".join(zones))
        return zones
    except ClientError as err:
        logger.error(
            "Couldn't get availability zones. Error: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def get_metrics(self, namespace: str, dimensions: list) -> list:
    """
    Gets a list of CloudWatch metrics filtered by namespace and dimensions.

    :param namespace: The namespace of the metrics to look up.
    :param dimensions: The dimensions of the metrics to look up.
    :return: The list of metrics.
    :raises ClientError: If there is an error retrieving CloudWatch metrics.
    """
    try:
        metrics = list(

```

```

        self.cloudwatch_resource.metrics.filter(
            Namespace=namespace, Dimensions=dimensions
        )
    )
    logger.info(
        "Retrieved metrics for namespace %s with dimensions %s.",
        namespace,
        dimensions,
    )
    return metrics
except ClientError as err:
    logger.error(
        "Couldn't get metrics for %s, %s. Error: %s: %s",
        namespace,
        dimensions,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

@staticmethod
def get_metric_statistics(
    dimensions: list, metric: str, start: datetime, end: datetime
) -> list:
    """
    Gets statistics for a CloudWatch metric within a specified time span.

    :param dimensions: The dimensions of the metric.
    :param metric: The metric to look up.
    :param start: The start of the time span for retrieved metrics.
    :param end: The end of the time span for retrieved metrics.
    :return: The list of data points found for the specified metric.
    :raises ClientError: If there is an error retrieving metric statistics.
    """
    try:
        response = metric.get_statistics(
            Dimensions=dimensions,
            StartTime=start,
            EndTime=end,
            Period=60,
            Statistics=["Sum"],
        )
        data = response["Datapoints"]
        logger.info("Retrieved statistics for metric %s.", metric.name)

```

```

        return data
    except ClientError as err:
        logger.error(
            "Couldn't get statistics for metric %s. Error: %s: %s",
            metric.name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def print_simplified_group(group: dict) -> None:
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired: "
        f"{group['DesiredCapacity']}"
    )
    if group["Instances"]:
        print(f"\tInstances:")
        for inst in group["Instances"]:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")

def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)

```



```
def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) ->
list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
    """
    ready = False
    instances = []
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids
    else []
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
            f"Here are the details of the instance{'s' if len(instances) > 1 else
''}:"
        )
        for instance in instances:
            pp(instance)
    return instance_ids
```

- 如需 API 詳細資訊，請參閱《適用於 Python (Boto3) 的 AWS SDK API 參考》中的下列主題。
- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)

- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
[package]
name = "autoscaling-code-examples"
version = "0.1.0"
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
  <dpsouth@amazon.com>"]
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
clap = { version = "4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"

use std::{collections::BTreeSet, fmt::Display};
```

```

use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:");
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "");
        }
        Ok(())
    }
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();

```

```

// 1. Create an EC2 launch template that you'll use to create an auto scaling
group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
template.
// 2. CreateAutoScalingGroup: pass it the launch template you created in step
0. Give it min/max of 1 instance.
// 4. EnableMetricsCollection: enable all metrics or a subset.
let scenario = match
AutoScalingScenario::prepare_scenario(&shared_config).await {
    Ok(scenario) => scenario,
    Err(errs) => {
        let err_str = errs
            .into_iter()
            .map(|e| e.to_string())
            .collect::<Vec<String>>()
            .join(", ");
        return Err( anyhow!("Failed to initialize scenario: {err_str}"));
    }
};

info!("Prepared autoscaling scenario:\n{scenario}");

let stable = scenario.wait_for_stable(1).await;
if let Err(err) = stable {
    warnings.push(
        "There was a problem while waiting for group to be stable",
        err,
    );
}

// 3. DescribeAutoScalingInstances: show that one instance has launched.
show_scenario_description(
    &scenario,
    "show that the group was created and one instance has launched",
)
.await;

// 5. UpdateAutoScalingGroup: update max size to 3.
let scale_max_size = scenario.scale_max_size(3).await;
if let Err(err) = scale_max_size {
    warnings.push("There was a problem scaling max size", err);
}

// 6. DescribeAutoScalingGroups: the current state of the group
show_scenario_description(

```

```

        &scenario,
        "show the current state of the group after setting max size",
    )
    .await;

// 7. SetDesiredCapacity: set desired capacity to 2.
let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
if let Err(err) = scale_desired_capacity {
    warnings.push("There was a problem setting desired capacity", err);
}

// Wait for a second instance to launch.
let stable = scenario.wait_for_stable(2).await;
if let Err(err) = stable {
    warnings.push(
        "There was a problem while waiting for group to be stable",
        err,
    );
}

// 8. DescribeAutoScalingInstances: show that two instances are launched.
show_scenario_description(
    &scenario,
    "show that two instances are launched after setting desired capacity",
)
.await;

let ids_before = scenario
    .list_instances()
    .await
    .map(|v| v.into_iter().collect::<BTreeSet<_>>())
    .unwrap_or_default();

// 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in
the group.
let terminate_some_instance = scenario.terminate_some_instance().await;
if let Err(err) = terminate_some_instance {
    warnings.push("There was a problem replacing an instance", err);
}

let wait_after_terminate = scenario.wait_for_stable(1).await;
if let Err(err) = wait_after_terminate {
    warnings.push(
        "There was a problem waiting after terminating an instance",

```

```

        err,
    );
}

let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
if let Err(err) = wait_scale_up_after_terminate {
    warnings.push(
        "There was a problem waiting for scale up after terminating an
instance",
        err,
    );
}

let ids_after = scenario
    .list_instances()
    .await
    .map(|v| v.into_iter().collect::<BTreeSet<_>>())
    .unwrap_or_default();

let difference = ids_after.intersection(&ids_before).count();
if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
    warnings.push(
        "Before and after set not different",
        ScenarioError::with(format!("{difference}")),
    );
}

// 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
show_scenario_description(
    &scenario,
    "list the scaling activities that have occurred for the group so far",
)
.await;

// 11. DisableMetricsCollection
let scale_group = scenario.scale_group_to_zero().await;
if let Err(err) = scale_group {
    warnings.push("There was a problem scaling the group to 0", err);
}
show_scenario_description(&scenario, "Scenario scaled to 0").await;

// 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):

```

```

// 13. Delete LaunchTemplate.
let clean_scenario = scenario.clean_scenario().await;
if let Err(errs) = clean_scenario {
    for err in errs {
        warnings.push("There was a problem cleaning the scenario", err);
    }
} else {
    info!("The scenario has been cleaned up!");
}

if warnings.is_empty() {
    Ok(())
} else {
    Err(anyhow!(
        "There were warnings during scenario execution:\n{warnings}"
    ))
}
}

pub mod scenario;

use std::{
    error::Error,
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
    "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
const AUTOSCALING_GROUP_NAME: &str =
    "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25
seconds.

```

```

const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at
a time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
            ))
        } else {
            tokio::time::sleep(WAIT_TIME).await;
            Ok(())
        }
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\\n",
            self.launch_template_arn
        ))
    }
}

```



```

    ))?;
    f.write_fmt(format_args!(
        "\tScaling Group Name: {}\n",
        self.auto_scaling_group_name
    ))?;

    Ok(())
}

pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t\t\tGroup status:")?;
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t\t- {status}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t\t! - {e}")?,
        }
        writeln!(f, "\t\t\tInstances:")?;
        match &self.instances {
            Ok(instances) => {
                for instance in instances {
                    writeln!(f, "\t\t\t- {instance}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t\t! {e}")?,
        }

        writeln!(f, "\t\t\tActivities:")?;
        match &self.activities {
            Ok(activities) => {
                for activity in activities {
                    writeln!(
                        f,
                        "\t\t\t- {} Progress: {}% Status: {:?} End: {:?}"

```

```

        activity.cause().unwrap_or("Unknown"),
        activity.progress.unwrap_or(-1),
        activity.status_code(),
        // activity.status_message().unwrap_or_default()
        activity.end_time(),
    )?;
    }
}
Err(e) => writeln!(f, "\t\t! {e}")?,
}

Ok(())
}
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(|s| s.to_string()),
            code: err.code().map(|s| s.to_string()),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("{code}"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug)]
pub struct ScenarioError {

```

```

    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
    // for this example
    // the ScenarioError captures the underlying information in MetadataError and
    // treats it as a
    // single Error from this Crate. In other contexts, it may be appropriate to
    // model the error
    // as including the SdkError as its source.
}

impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
Vec<ScenarioError>> {
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);
    }
}

```

```

let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

// Before creating any resources, prepare the list of AZs
let availability_zones = ec2.describe_availability_zones().send().await;
if let Err(err) = availability_zones {
    return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
}

let availability_zones: Vec<String> = availability_zones
    .unwrap()
    .availability_zones
    .unwrap_or_default()
    .iter()
    .take(3)
    .map(|z| z.zone_name.clone().unwrap())
    .collect();

// 1. Create an EC2 launch template that you'll use to create an auto
scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
template.
// * Recommended: InstanceType='t1.micro',
ImageId='ami-0ca285d4c2cda3300'
let create_launch_template = ec2
    .create_launch_template()
    .launch_template_name(LAUNCH_TEMPLATE_NAME)
    .launch_template_data(
        RequestLaunchTemplateData::builder()
            .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
            .image_id("ami-0ca285d4c2cda3300")
            .build(),
    )
    .send()
    .await
    .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)]?);

let launch_template_arn = match create_launch_template.launch_template {
    Some(launch_template) =>
launch_template.launch_template_id.unwrap_or_default(),
    None => {
        // Try to delete the launch template
        let _ = ec2
            .delete_launch_template()

```

```

        .launch_template_name(LAUNCH_TEMPLATE_NAME)
        .send()
        .await;
    return Err(vec![ScenarioError::with("Failed to load launch
template")]));
    }
};

// 2. CreateAutoScalingGroup: pass it the launch template you created in
step 0. Give it min/max of 1 instance.
// You can use EC2.describe_availability_zones() to get a list of AZs
(you have to specify an AZ when you create the group).
// Wait for instance to launch. Use a waiter if you have one, otherwise
DescribeAutoScalingInstances until LifecycleState='InService'
if let Err(err) = autoscaling
    .create_auto_scaling_group()
    .auto_scaling_group_name(auto_scaling_group_name.as_str())
    .launch_template(
        LaunchTemplateSpecification::builder()
            .launch_template_id(launch_template_arn.clone())
            .version("$Latest")
            .build(),
    )
    .max_size(1)
    .min_size(1)
    .set_availability_zones(Some(availability_zones))
    .send()
    .await
{
    let mut errs = vec![ScenarioError::new(
        "Failed to create autoscaling group",
        &err,
    )];

    if let Err(err) = autoscaling
        .delete_auto_scaling_group()
        .auto_scaling_group_name(auto_scaling_group_name.as_str())
        .send()
        .await
    {
        errs.push(ScenarioError::new(
            "Failed to clean up autoscaling group",
            &err,
        ));
    }
}

```

```

    }

    if let Err(err) = ec2
        .delete_launch_template()
        .launch_template_id(launch_template_arn.clone())
        .send()
        .await
    {
        errs.push(ScenarioError::new(
            "Failed to clean up launch template",
            &err,
        ));
    }
    return Err(errs);
}

let scenario = AutoScalingScenario {
    ec2,
    autoscaling: autoscaling.clone(), // Clients are cheap so cloning
here to prevent a move is ok.
    auto_scaling_group_name: auto_scaling_group_name.clone(),
    launch_template_arn,
};

let enable_metrics_collection = autoscaling
    .enable_metrics_collection()
    .auto_scaling_group_name(auto_scaling_group_name.as_str())
    .granularity("1Minute")
    .set_metrics(Some(vec![
        String::from("GroupMinSize"),
        String::from("GroupMaxSize"),
        String::from("GroupDesiredCapacity"),
        String::from("GroupInServiceInstances"),
        String::from("GroupTotalInstances"),
    ]))
    .send()
    .await;

match enable_metrics_collection {
    Ok(_) => Ok(scenario),
    Err(err) => {
        scenario.clean_scenario().await?;
        Err(vec![ScenarioError::new(
            "Failed to enable metrics collections for group",

```

```

        &err,
    )])
    }
}

pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
    let _ = self.wait_for_no_scaling().await;
    let delete_group = self
        .autoscaling
        .delete_auto_scaling_group()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .send()
        .await;

    // 14. Delete LaunchTemplate.
    let delete_launch_template = self
        .ec2
        .delete_launch_template()
        .launch_template_id(self.launch_template_arn.clone())
        .send()
        .await;

    let early_exit = match (delete_group, delete_launch_template) {
        (Ok(_), Ok(_)) => Ok(()),
        (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
            "There was an error cleaning the launch template",
            &e,
        )]),
        (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
            "There was an error cleaning the scale group",
            &e,
        )]),
        (Err(e1), Err(e2)) => Err(vec![
            ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
            ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
        ]),
    };

    if early_exit.is_err() {
        early_exit
    } else {

```

```

        // Wait for delete_group to finish
        let waiter = Waiter::new();
        let mut errors = Vec::<ScenarioError>::new();
        while errors.len() < 3 {
            if let Err(e) = waiter.sleep().await {
                errors.push(e);
                continue;
            }
            let describe_group = self
                .autoscaling
                .describe_auto_scaling_groups()

        .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;
            match describe_group {
                Ok(group) => match group.auto_scaling_groups().first() {
                    Some(group) => {
                        if group.status() != Some("Delete in progress") {
                            errors.push(ScenarioError::with(format!(
                                "Group in an unknown state while deleting:
{}\"",
                                group.status().unwrap_or("unknown error")
                            )));
                            return Err(errors);
                        }
                    }
                    None => return Ok(()),
                },
                Err(err) => {
                    errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
                }
            }
            if errors.len() > 3 {
                return Err(errors);
            }
        }
        Err(vec![ScenarioError::with(
            "Exited cleanup wait loop without retuning success or failing
after three rounds",
        )])
    }
}

```



```

pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
    let group = self
        .autoscaling
        .describe_auto_scaling_groups()
        .auto_scaling_group_names(self.auto_scaling_group_name.clone())
        .send()
        .await
        .map(|s| {
            s.auto_scaling_groups()
                .iter()
                .map(|s| {
                    format!(
                        "{}: {}",
                        s.auto_scaling_group_name().unwrap_or("Unknown"),
                        s.status().unwrap_or("Unknown")
                    )
                })
                .collect::<Vec<String>>()
        })
        .map_err(|e| {
            ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
        });

    let instances = self
        .list_instances()
        .await
        .map_err(|e| anyhow!("There was an error listing instances: {e}",));

    // 10. DescribeScalingActivities: list the scaling activities that have
    occurred for the group so far.
    // Bonus: use CloudWatch API to get and show some metrics collected for
    the group.
    // CW.ListMetrics with Namespace='AWS/AutoScaling' and
    Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
    // CW.GetMetricStatistics with Statistics='Sum'. Start and End times
    must be in UTC!
    let activities = self
        .autoscaling
        .describe_scaling_activities()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .into_paginator()
        .items()

```

```

        .send()
        .collect::()
        .await
        .map_err(|e| {
            anyhow!(
                "There was an error retrieving scaling activities: {}",
                DisplayErrorContext(&e)
            )
        });

    AutoScalingScenarioDescription {
        group,
        instances,
        activities,
    }
}

async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
    let describe_auto_scaling_groups = self
        .autoscaling
        .describe_auto_scaling_groups()
        .auto_scaling_group_names(self.auto_scaling_group_name.clone())
        .send()
        .await;

    if let Err(err) = describe_auto_scaling_groups {
        return Err(ScenarioError::new(
            format!(
                "Failed to get status of autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )
            .as_str(),
            &err,
        ));
    }

    let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
    let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
    let auto_scaling_group = auto_scaling_groups.first();

    if auto_scaling_group.is_none() {
        return Err(ScenarioError::with(format!(

```

```

        "Could not find autoscaling group {}",
        self.auto_scaling_group_name.clone()
    ));
}

Ok(auto_scaling_group.unwrap().clone())
}

pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
    let waiter = Waiter::new();
    let mut scaling = true;
    while scaling {
        waiter.sleep().await?;
        let describe_activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map_err(|e| {
                ScenarioError::new("Failed to get autoscaling activities for
group", &e)
            })?;
        let activities = describe_activities.activities();
        trace!(
            "Waiting for no scaling found {} activities",
            activities.len()
        );
        scaling = activities.iter().any(|a| a.progress() < Some(100));
    }
    Ok(())
}

pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError>
{
    self.wait_for_no_scaling().await?;

    let mut group = self.get_group().await?;
    let mut count = count_group_instances(&group);

    let waiter = Waiter::new();
    while count != size {
        trace!("Waiting for stable {size} (current: {count})");
        waiter.sleep().await?;
    }
}

```

```

        group = self.get_group().await?;
        count = count_group_instances(&group);
    }

    Ok(())
}

pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
    // The direct way to list instances is by using
    DescribeAutoScalingGroup's instances property. However, this returns a
    Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
    // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

    // Alternatively, and for the sake of example,
    DescribeAutoScalingInstances returns a list that can be filtered by the client.
    self.autoscaling
        .describe_auto_scaling_instances()
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await
        .map(|items| {
            items
                .into_iter()
                .filter(|i| {
                    i.auto_scaling_group_name.as_deref()
                        == Some(self.auto_scaling_group_name.as_str())
                })
                .map(|i| i.instance_id.unwrap_or_default())
                .filter(|id| !id.is_empty())
                .collect::<Vec<String>>()
        })
        .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }

    pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())

```

```

        .min_size(size)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            format!("Failer to update group to min size ({size}))").as_str(),
            &err,
        ));
    }
    Ok(())
}

pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
    // 5. UpdateAutoScalingGroup: update max size to 3.
    let update_group = self
        .autoscaling
        .update_auto_scaling_group()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .max_size(size)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            format!("Failed to update group to max size ({size}))").as_str(),
            &err,
        ));
    }
    Ok(())
}

pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
    // 7. SetDesiredCapacity: set desired capacity to 2.
    // Wait for a second instance to launch.
    let update_group = self
        .autoscaling
        .set_desired_capacity()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .desired_capacity(capacity)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(

```

```

        format!("Failed to update group to desired capacity
({capacity}))").as_str(),
        &err,
    ));
}
Ok(())
}

pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
    // If this fails it's fine, just means there are extra cloudwatch metrics
    events for the scale-down.
    let _ = self
        .autoscaling
        .disable_metrics_collection()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .send()
        .await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all
    instances):
    // UpdateAutoScalingGroup with MinSize=0
    let update_group = self
        .autoscaling
        .update_auto_scaling_group()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .min_size(0)
        .desired_capacity(0)
        .send()
        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            "Failed to update group for scaling down&",
            &err,
        ));
    }

    let stable = self.wait_for_stable(0).await;
    if let Err(err) = stable {
        return Err(ScenarioError::with(format!(
            "Error while waiting for group to be stable on scale down: {err}"
        )));
    }

    Ok(())
}

```

```

    }

    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Rust API reference 中的下列主題。
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)

- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDKs 的 Auto Scaling 動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 Auto Scaling 動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

這些摘錄會呼叫 Auto Scaling API，是必須在內容中執行之大型程式的程式碼摘錄。您可以在 [使用 AWS SDKs 的 Auto Scaling 案例](#) 中查看內容中的動作。

下列範例僅包含最常使用的動作。如需完整清單，請參閱《[Amazon EC2 Auto Scaling API 參考](#)》。

範例

- [搭配使用 AttachInstances 與 CLI](#)
- [AttachLoadBalancerTargetGroups 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 AttachLoadBalancers 與 CLI](#)
- [搭配使用 CompleteLifecycleAction 與 CLI](#)
- [CreateAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 CreateLaunchConfiguration 與 CLI](#)
- [搭配使用 CreateOrUpdateTags 與 CLI](#)
- [DeleteAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 DeleteLaunchConfiguration 與 CLI](#)
- [搭配使用 DeleteLifecycleHook 與 CLI](#)
- [搭配使用 DeleteNotificationConfiguration 與 CLI](#)

- [搭配使用 DeletePolicy 與 CLI](#)
- [搭配使用 DeleteScheduledAction 與 CLI](#)
- [搭配使用 DeleteTags 與 CLI](#)
- [搭配使用 DescribeAccountLimits 與 CLI](#)
- [搭配使用 DescribeAdjustmentTypes 與 CLI](#)
- [DescribeAutoScalingGroups 搭配 AWS SDK 或 CLI 使用](#)
- [DescribeAutoScalingInstances 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 DescribeAutoScalingNotificationTypes 與 CLI](#)
- [搭配使用 DescribeLaunchConfigurations 與 CLI](#)
- [搭配使用 DescribeLifecycleHookTypes 與 CLI](#)
- [搭配使用 DescribeLifecycleHooks 與 CLI](#)
- [搭配使用 DescribeLoadBalancers 與 CLI](#)
- [搭配使用 DescribeMetricCollectionTypes 與 CLI](#)
- [搭配使用 DescribeNotificationConfigurations 與 CLI](#)
- [搭配使用 DescribePolicies 與 CLI](#)
- [DescribeScalingActivities 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 DescribeScalingProcessTypes 與 CLI](#)
- [搭配使用 DescribeScheduledActions 與 CLI](#)
- [搭配使用 DescribeTags 與 CLI](#)
- [搭配使用 DescribeTerminationPolicyTypes 與 CLI](#)
- [搭配使用 DetachInstances 與 CLI](#)
- [搭配使用 DetachLoadBalancers 與 CLI](#)
- [DisableMetricsCollection 搭配 AWS SDK 或 CLI 使用](#)
- [EnableMetricsCollection 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 EnterStandby 與 CLI](#)
- [搭配使用 ExecutePolicy 與 CLI](#)
- [搭配使用 ExitStandby 與 CLI](#)
- [搭配使用 PutLifecycleHook 與 CLI](#)
- [搭配使用 PutNotificationConfiguration 與 CLI](#)

- [搭配使用 PutScalingPolicy 與 CLI](#)
- [搭配使用 PutScheduledUpdateGroupAction 與 CLI](#)
- [搭配使用 RecordLifecycleActionHeartbeat 與 CLI](#)
- [搭配使用 ResumeProcesses 與 CLI](#)
- [SetDesiredCapacity 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 SetInstanceHealth 與 CLI](#)
- [搭配使用 SetInstanceProtection 與 CLI](#)
- [搭配使用 SuspendProcesses 與 CLI](#)
- [TerminateInstanceInAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)
- [UpdateAutoScalingGroup 搭配 AWS SDK 或 CLI 使用](#)

搭配使用 **AttachInstances** 與 CLI

下列程式碼範例示範如何使用 `AttachInstances`。

CLI

AWS CLI

將執行個體連接至 Auto Scaling 群組

此範例將指定的執行個體連接至指定的 Auto Scaling 群組。

```
aws autoscaling attach-instances \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [AttachInstances](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的執行個體連接至指定的 Auto Scaling 群組。Auto Scaling 會自動增加 Auto Scaling 群組所需的容量。

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [AttachInstances](#)。

Tools for PowerShell V5

範例 1：此範例將指定的執行個體連接至指定的 Auto Scaling 群組。Auto Scaling 會自動增加 Auto Scaling 群組所需的容量。

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [AttachInstances](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

AttachLoadBalancerTargetGroups 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 AttachLoadBalancerTargetGroups。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [建置及管理彈性服務](#)

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>
```

```

    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
    Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [AttachLoadBalancerTargetGroups](#)。

CLI

AWS CLI

將目標群組連接到 Auto Scaling 群組

此範例將指定的目標群組連接至指定的 Auto Scaling 群組。

```

aws autoscaling attach-load-balancer-target-groups \
    --auto-scaling-group-name my-asg \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067

```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Elastic Load Balancing](#) 和 [Amazon EC2 Auto Scaling](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [AttachLoadBalancerTargetGroups](#)。

JavaScript

適用於 JavaScript (v3) 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
const client = new AutoScalingClient({});
await client.send(
  new AttachLoadBalancerTargetGroupsCommand({
    AutoScalingGroupName: NAMES.autoScalingGroupName,
    TargetGroupARNs: [state.targetGroupArn],
  }),
);
```

- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的 [AttachLoadBalancerTargetGroups](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
```

```

    inst_type: str,
    ami_param: str,
    autoscaling_client: boto3.client,
    ec2_client: boto3.client,
    ssm_client: boto3.client,
    iam_client: boto3.client,
):
    """
    Initializes the AutoScaler class with the necessary parameters.

    :param resource_prefix: The prefix for naming AWS resources that are
    created by this class.
    :param inst_type: The type of EC2 instance to create, such as t3.micro.
    :param ami_param: The Systems Manager parameter used to look up the AMI
    that is created.
    :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
    :param ec2_client: A Boto3 EC2 client.
    :param ssm_client: A Boto3 Systems Manager client.
    :param iam_client: A Boto3 IAM client.
    """
    self.inst_type = inst_type
    self.ami_param = ami_param
    self.autoscaling_client = autoscaling_client
    self.ec2_client = ec2_client
    self.ssm_client = ssm_client
    self.iam_client = iam_client
    sts_client = boto3.client("sts")
    self.account_id = sts_client.get_caller_identity()["Account"]

    self.key_pair_name = f"{resource_prefix}-key-pair"
    self.launch_template_name = f"{resource_prefix}-template-"
    self.group_name = f"{resource_prefix}-group"

    # Happy path
    self.instance_policy_name = f"{resource_prefix}-pol"
    self.instance_role_name = f"{resource_prefix}-role"
    self.instance_profile_name = f"{resource_prefix}-prof"

    # Failure mode
    self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
    self.bad_creds_role_name = f"{resource_prefix}-bc-role"
    self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

```

```

def attach_load_balancer_target_group(
    self, lb_target_group: Dict[str, Any]
) -> None:
    """
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
    Scaling group.
    The target group specifies how the load balancer forwards requests to the
    instances
    in the group.

    :param lb_target_group: Data about the ELB target group to attach.
    """
    try:
        self.autoscaling_client.attach_load_balancer_target_groups(
            AutoScalingGroupName=self.group_name,
            TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
        )
        log.info(
            "Attached load balancer target group %s to auto scaling group
%s.",
            lb_target_group["TargetGroupName"],
            self.group_name,
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
        )
        if error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the resource."
            )
        elif error_code == "ServiceLinkedRoleFailure":
            log.error(
                "The operation failed because the service-linked role is not
ready or does not exist. "
                "Check that the service-linked role exists and is correctly
configured."
            )
        log.error(f"Full error:\n\t{err}")

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [AttachLoadBalancerTargetGroups](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **AttachLoadBalancers** 與 CLI

下列程式碼範例示範如何使用 AttachLoadBalancers。

CLI

AWS CLI

將 Classic Load Balancer 連接至 Auto Scaling 群組

此範例將指定的 Classic Load Balancer 連接至指定的 Auto Scaling 群組。

```
aws autoscaling attach-load-balancers \  
  --load-balancer-names my-load-balancer \  
  --auto-scaling-group-name my-asg
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Elastic Load Balancing](#) 和 [Amazon EC2 Auto Scaling](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [AttachLoadBalancers](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的負載平衡器連接至指定的 Auto Scaling 群組。

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [AttachLoadBalancers](#)。

Tools for PowerShell V5

範例 1：此範例將指定的負載平衡器連接至指定的 Auto Scaling 群組。

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [AttachLoadBalancers](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **CompleteLifecycleAction** 與 CLI

下列程式碼範例示範如何使用 CompleteLifecycleAction。

CLI

AWS CLI

完成生命週期動作

此範例通知 Amazon EC2 Auto Scaling 指定的生命週期動作已完成，因此可以完成啟動或終止執行個體。

```
aws autoscaling complete-lifecycle-action \  
  --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-action-result CONTINUE \  
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling lifecycle hook](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CompleteLifecycleAction](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例會完成指定的生命週期動作。

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -  
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [CompleteLifecycleAction](#)。

Tools for PowerShell V5

範例 1：此範例會完成指定的生命週期動作。

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -  
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [CompleteLifecycleAction](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

CreateAutoScalingGroup 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateAutoScalingGroup。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)
- [建置及管理彈性服務](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
    {
        availabilityZone,
    };

    var request = new CreateAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        AvailabilityZones = zoneList,
        LaunchTemplate = templateSpecification,
        MaxSize = 6,
        MinSize = 1
    };
};
```

```
try
{
    var response = await
        _amazonAutoScaling.CreateAutoScalingGroupAsync(request);
    Console.WriteLine($"{groupName} Auto Scaling Group created");
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AlreadyExistsException)
{
    Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
    return true;
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [CreateAutoScalingGroup](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);
Aws::Vector<Aws::String> availabilityGroupZones;
availabilityGroupZones.push_back(
    availabilityZones[availabilityZoneChoice - 1].GetZoneName());
request.SetAvailabilityZones(availabilityGroupZones);
```

```

        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
            autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
            Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }
}

```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [CreateAutoScalingGroup](#)。

CLI

AWS CLI

範例 1：建立 Auto Scaling 群組

下列 create-auto-scaling-group 範例會在區域內多個可用區域的子網路中建立 Auto Scaling 群組。執行個體會以指定啟動範本的預設版本啟動。請注意，預設值用於大多數其他設定，例如終止政策和運作狀態檢查組態。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12 \  
  --min-size 1 \  
  --max-size 5 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Auto Scaling 群組](#)。

範例 2：附加 Application Load Balancer、Network Load Balancer 或 Gateway Load Balancer。

此範例為支援預期流量的負載平衡器指定目標群組的 ARN。運作狀態檢查類型會指定 ELB，因此當 Elastic Load Balancing 將執行個體報告為運作狀態不佳時，Auto Scaling 群組會取而代之。該命令也會定義 600 秒的運作狀態檢查寬限期。寬限期有助於防止新啟動的執行個體提早終止。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12 \  
  --target-group-arns arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-targets/943f017f100becff \  
  --health-check-type ELB \  
  --health-check-grace-period 600 \  
  --min-size 1 \  
  --max-size 5 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Elastic Load Balancing](#) 和 [Amazon EC2 Auto Scaling](#)。

範例 3：指定置放群組並，使用最新版本的啟動範本

此範例在單一可用區域內的置放群組中啟動執行個體。這對於具有 HPC 工作負載的低延遲群組很有用。此範例也會指定群組的大小下限、上限和所需容量。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --min-size 1 \  
  --max-size 5 \  
  --desired-capacity 3 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

```
--launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest' \  
--min-size 1 \  
--max-size 5 \  
--desired-capacity 3 \  
--placement-group my-placement-group \  
--vpc-zone-identifier "subnet-6194ea3b"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的[置放群組](#)。

範例 4：指定單一執行個體 Auto Scaling 群組，並使用特定版本的啟動範本

此範例建立一個容量上下限設為 1 的 Auto Scaling 群組，以強制執行一個執行個體。該命令也指定啟動範本的 v1，其中會指定現有 ENI 的 ID。當您使用指定 eth0 之現有 ENI 的啟動範本時，必須為符合網路介面的 Auto Scaling 群組指定可用區域，無需另外在請求中指定子網路 ID。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg-single-instance \  
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1' \  
  \  
  --min-size 1 \  
  --max-size 1 \  
  --availability-zones us-west-2a
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[Auto Scaling 群組](#)。

範例 5：指定不同的終止政策

此範例使用啟動組態建立 Auto Scaling 群組，並將終止政策設定為先終止最舊的執行個體。該命令也會將標籤套用至群組及其執行個體，索引鍵為 Role，值為 WebServer。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-configuration-name my-lc \  
  --min-size 1 \  
  --max-size 5 \  
  --termination-policies "OldestInstance" \  
  --tags "ResourceId=my-asg,ResourceType=auto-scaling-  
group,Key=Role,Value=WebServer,PropagateAtLaunch=true" \  
  \  
  --availability-zones us-west-2a
```

```
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用 Amazon EC2 Auto Scaling 終止政策](#)。

範例 6：指定啟動 lifecycle hook

此範例會建立具有在執行個體啟動時支援自訂動作之 lifecycle hook 的 Auto Scaling 群組。

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

config.json 檔案的內容：

```
{  
  "AutoScalingGroupName": "my-asg",  
  "LaunchTemplate": {  
    "LaunchTemplateId": "lt-1234567890abcde12"  
  },  
  "LifecycleHookSpecificationList": [{  
    "LifecycleHookName": "my-launch-hook",  
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",  
    "NotificationTargetARN": "arn:aws:sqs:us-west-2:123456789012:my-sqs-  
queue",  
    "RoleARN": "arn:aws:iam::123456789012:role/my-notification-role",  
    "NotificationMetadata": "SQS message metadata",  
    "HeartbeatTimeout": 4800,  
    "DefaultResult": "ABANDON"  
  }],  
  "MinSize": 1,  
  "MaxSize": 5,  
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",  
  "Tags": [{  
    "ResourceType": "auto-scaling-group",  
    "ResourceId": "my-asg",  
    "PropagateAtLaunch": true,  
    "Value": "test",  
    "Key": "environment"  
  }]  
}
```


此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling lifecycle hook](#)。

範例 7：指定終止 lifecycle hook

此範例會建立具有在執行個體終止時支援自訂動作之 lifecycle hook 的 Auto Scaling 群組。

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

config.json 的內容：

```
{  
  "AutoScalingGroupName": "my-asg",  
  "LaunchTemplate": {  
    "LaunchTemplateId": "lt-1234567890abcde12"  
  },  
  "LifecycleHookSpecificationList": [{  
    "LifecycleHookName": "my-termination-hook",  
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",  
    "HeartbeatTimeout": 120,  
    "DefaultResult": "CONTINUE"  
  }],  
  "MinSize": 1,  
  "MaxSize": 5,  
  "TargetGroupARNs": [  
    "arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-  
targets/73e2d6bc24d8a067"  
  ],  
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"  
}
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling lifecycle hook](#)。

範例 8：指定自訂終止政策

此範例會建立 Auto Scaling 群組，指定自訂 Lambda 函式終止政策，告知 Amazon EC2 Auto Scaling 哪些執行個體可以在縮減時安全地終止。

```
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name my-asg-single-instance \
  --launch-template LaunchTemplateName=my-template-for-auto-scaling \
  --min-size 1 \
  --max-size 5 \
  --termination-policies "arn:aws:lambda:us-west-2:123456789012:function:HelloFunction:prod" \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用 Lambda 建立自訂終止政策](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[CreateAutoScalingGroup](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
  software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
  software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
  software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
  software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
```

```

* Before running this SDK for Java (v2) code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <serviceLinkedRoleARN>
<vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
                vpcZoneId - A subnet Id for a virtual private cloud (VPC)
where instances in the Auto Scaling group can be created.
            "";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
        String groupName,
        String launchTemplateName,

```

```
String vpcZoneId) {

    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
            .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [CreateAutoScalingGroup](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun createAutoScalingGroup(  
    groupName: String,  
    launchTemplateNameVal: String,  
    serviceLinkedRoleARNVal: String,  
    vpcZoneIdVal: String,  
) {  
    val templateSpecification =  
        LaunchTemplateSpecification {  
            launchTemplateName = launchTemplateNameVal  
        }  
  
    val request =  
        CreateAutoScalingGroupRequest {  
            autoScalingGroupName = groupName  
            availabilityZones = listOf("us-east-1a")  
            launchTemplate = templateSpecification  
            maxSize = 1  
            minSize = 1  
            vpcZoneIdentifier = vpcZoneIdVal  
            serviceLinkedRoleArn = serviceLinkedRoleARNVal  
        }  
  
    // This object is required for the waiter call.  
    val groupsRequestWaiter =  
        DescribeAutoScalingGroupsRequest {  
            autoScalingGroupNames = listOf(groupName)  
        }  
  
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
        autoScalingClient.createAutoScalingGroup(request)  
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)  
        println("$groupName was created!")  
    }
```

```
}  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [CreateAutoScalingGroup](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function createAutoScalingGroup(  
    $autoScalingGroupName,  
    $availabilityZones,  
    $minSize,  
    $maxSize,  
    $launchTemplateId  
) {  
    return $this->autoScalingClient->createAutoScalingGroup([  
        'AutoScalingGroupName' => $autoScalingGroupName,  
        'AvailabilityZones' => $availabilityZones,  
        'MinSize' => $minSize,  
        'MaxSize' => $maxSize,  
        'LaunchTemplate' => [  
            'LaunchTemplateId' => $launchTemplateId,  
        ],  
    ]);  
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [CreateAutoScalingGroup](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例使用指定的名稱及屬性建立 Auto Scaling 群組。預設所需容量為大小下限。因此，此 Auto Scaling 群組會啟動兩個執行個體，其中一個位於指定的兩個可用區域中。

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [CreateAutoScalingGroup](#)。

Tools for PowerShell V5

範例 1：此範例使用指定的名稱及屬性建立 Auto Scaling 群組。預設所需容量為大小下限。因此，此 Auto Scaling 群組會啟動兩個執行個體，其中一個位於指定的兩個可用區域中。

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [CreateAutoScalingGroup](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
```

```

        self.autoscaling_client = autoscaling_client

    def create_group(
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
    ) -> None:
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
        :param group_zones: The Availability Zones in which instances can be
        created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
        template.
                                The launch template specifies the
        configuration of
                                instances that are created by auto scaling
        activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling
        group.
        """
        try:
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=group_name,
                AvailabilityZones=group_zones,
                LaunchTemplate={
                    "LaunchTemplateName": launch_template_name,
                    "Version": "$Default",
                },
                MinSize=min_size,
                MaxSize=max_size,
            )

            # Wait for the group to exist.
            waiter = self.autoscaling_client.get_waiter("group_exists")
            waiter.wait(AutoScalingGroupNames=[group_name])

```



```

        logger.info(f"Successfully created Auto Scaling group {group_name}.")

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(f"Failed to create Auto Scaling group {group_name}.")
        if error_code == "AlreadyExistsFault":
            logger.error(
                f"An Auto Scaling group with the name '{group_name}' already
exists. "
                "Please use a different name or update the existing group.",
            )
        elif error_code == "LimitExceededFault":
            logger.error(
                "The request failed because you have reached the limit "
                "on the number of Auto Scaling groups or launch
configurations. "
                "Consider deleting unused resources or request a limit
increase. "
                "\nSee Auto Scaling Service Quota documentation here:"
                "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/
ec2-auto-scaling-quotas.html"
            )
            logger.error(f"Full error:\n\t{err}")
            raise

```

- 如需 API 詳細資訊，請參閱《適用於 Python (Boto3) 的 AWS SDK API 參考》中的 [CreateAutoScalingGroup](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error>
{
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}

```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的 AWS SDK API 參考》中的 [CreateAutoScalingGroup](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
DATA lo_launch_template TYPE REF TO /aws1/cl_aslaunchtemplatespec.
```

```
" Example: iv_group_name = 'my-auto-scaling-group'
```

```
" Example: iv_launch_template_name = 'my-launch-template'
```

```
" Example: iv_min_size = 1
```

```
" Example: iv_max_size = 3
```

```
" Example: iv_vpc_zone_identifier = 'subnet-12345,subnet-67890' (for VPC)
```

```
TRY.
```

```
" Create launch template specification
```

```
lo_launch_template = NEW /aws1/cl_aslaunchtemplatespec(
```

```

        iv_launchtemplatename = iv_launch_template_name
        iv_version = '$Default' ).

" Create the Auto Scaling group
" Use VPCZoneIdentifier for VPC subnets, or AvailabilityZones for EC2-
Classic
IF iv_vpc_zone_identifier IS NOT INITIAL.
    " VPC-based deployment - use subnet IDs
    ao_asc->createautoscalinggroup(
        iv_autoscalinggroupname = iv_group_name
        iv_vpczoneidentifier = iv_vpc_zone_identifier
        io_launchtemplate = lo_launch_template
        iv_minsize = iv_min_size
        iv_maxsize = iv_max_size ).
ELSE.
    " EC2-Classic or default VPC - use availability zones
    ao_asc->createautoscalinggroup(
        iv_autoscalinggroupname = iv_group_name
        it_availabilityzones = it_group_zones
        io_launchtemplate = lo_launch_template
        iv_minsize = iv_min_size
        iv_maxsize = iv_max_size ).
ENDIF.

" Wait for the group to be created (simplified - in production use proper
polling)
WAIT UP TO 10 SECONDS.

MESSAGE 'Auto Scaling group created successfully' TYPE 'I'.

CATCH /aws1/cx_ascalreadyexistsfault INTO DATA(lo_already_exists).
    RAISE EXCEPTION lo_already_exists.
CATCH /aws1/cx_asclimitexceededfault INTO DATA(lo_limit_exceeded).
    RAISE EXCEPTION lo_limit_exceeded.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [CreateAutoScalingGroup](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 `CreateLaunchConfiguration` 與 CLI

下列程式碼範例示範如何使用 `CreateLaunchConfiguration`。

CLI

AWS CLI

範例 1：建立啟動組態。

此範例會建立簡單的啟動組態。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[建立啟動組態](#)。

範例 2：使用安全群組、金鑰對和引導指令碼建立啟動組態

此範例會建立啟動組態，其中包含安全群組、金鑰對和使用者資料中包含的引導指令碼。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --security-groups sg-eb2af88example \  
  --key-name my-key-pair \  
  --user-data file://myuserdata.txt
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[建立啟動組態](#)。

範例 3：使用 IAM 角色建立啟動組態

此範例使用 IAM 角色的執行個體設定檔名稱建立啟動組態。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --iam-instance-profile my-autoscaling-role
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[在 Amazon EC2 執行個體上執行的應用程式 IAM 角色](#)。

範例 4：建立啟用詳細監控的啟動組態

此範例建立啟用 EC2 詳細監控的啟動組態，並在 1 分鐘內將 EC2 指標傳送至 CloudWatch。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --instance-monitoring Enabled=true
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[設定監控 Auto Scaling 執行個體](#)。

範例 5：建立啟動 Spot 執行個體的啟動組態

此範例建立使用 Spot 執行個體做為唯一購買選項的啟動組態。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --spot-price "0.50"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[請求 Spot 執行個體](#)。

範例 6：使用 EC2 執行個體建立啟動組態

此範例根據現有執行個體的屬性建立啟動組態。這麼做會覆寫置放租用，以及是否藉由納入 `--placement-tenancy` 和 `--no-associate-public-ip-address` 選項來設定公有 IP 位址。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc-from-instance \  
  --instance-id i-0123a456700123456 \  
  --instance-type m5.large \  
  --no-associate-public-ip-address \  
  --placement-tenancy dedicated
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用 EC2 執行個體建立啟動組態](#)。

範例 7：使用 Amazon EBS 磁碟區的區塊型儲存設備映射建立啟動組態

此範例為裝置名稱 `/dev/sdh` 且磁碟區大小為 20 的 Amazon EBS gp3 磁碟區，建立具有區塊型儲存設備映射的啟動組態。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --block-device-mappings '[{"DeviceName":"/dev/sdh", "Ebs":  
  {"VolumeSize":20, "VolumeType":"gp3"}}]'
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的[EBS](#)。

如需引用 JSON 格式參數值的語法資訊，請參閱《AWS 命令列界面使用者指南》中的[使用引號搭配 CLI AWS 中的字串](#)。

範例 8：使用執行個體儲存體磁碟區的區塊型儲存設備映射建立啟動組態

此範例使用 `ephemeral1` 建立啟動組態，做為裝置名稱 `/dev/sdc` 的執行個體儲存體磁碟區。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --instance-store-configuration '[{"DeviceName":"/dev/sdc", "EphemeralName": "ephemeral1"}]'
```

```
--image-id ami-04d5cc9b88example \  
--instance-type m5.large \  
--block-device-mappings '[{"DeviceName":"/dev/  
sdc", "VirtualName":"ephemeral1"}]'
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [BlockDeviceMapping](#)。

如需引用 JSON 格式參數值的語法資訊，請參閱《AWS 命令列界面使用者指南》中的[使用引號搭配 CLI AWS 中的字串](#)。

範例 9：建立啟動組態，並禁止區塊型儲存設備在啟動時連接

此範例會建立啟動組態，以隱藏 AMI 的區塊型儲存設備映射所指定的區塊型儲存設備 (例如，/dev/sdf)。

```
aws autoscaling create-launch-configuration \  
--launch-configuration-name my-lc \  
--image-id ami-04d5cc9b88example \  
--instance-type m5.large \  
--block-device-mappings '[{"DeviceName":"/dev/sdf", "NoDevice":""}]'
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [BlockDeviceMapping](#)。

如需引用 JSON 格式參數值的語法資訊，請參閱《AWS 命令列界面使用者指南》中的[使用引號搭配 CLI AWS 中的字串](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CreateLaunchConfiguration](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例建立名為 'my-lc' 的啟動組態。使用此啟動組態的 Auto Scaling 群組啟動的 EC2 執行個體，會使用指定的執行個體類型、AMI、安全群組和 IAM 角色。

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType  
"m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -  
IamInstanceProfile "myIamRole"
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [CreateLaunchConfiguration](#)。

Tools for PowerShell V5

範例 1：此範例建立名為 'my-lc' 的啟動組態。使用此啟動組態的 Auto Scaling 群組啟動的 EC2 執行個體，會使用指定的執行個體類型、AMI、安全群組和 IAM 角色。

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType  
"m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -  
IamInstanceProfile "myIamRole"
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [CreateLaunchConfiguration](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **CreateOrUpdateTags** 與 CLI

下列程式碼範例示範如何使用 CreateOrUpdateTags。

CLI

AWS CLI

為 Auto Scaling 群組中建立或更新標籤

此範例將兩個標籤新增至指定的 Auto Scaling 群組。

```
aws autoscaling create-or-update-tags \  
  --tags ResourceId=my-asg,ResourceType=auto-scaling-  
group,Key=Role,Value=WebServer,PropagateAtLaunch=true ResourceId=my-  
asg,ResourceType=auto-scaling-  
group,Key=Dept,Value=Research,PropagateAtLaunch=true
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [標記 Auto Scaling 群組和執行個體](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CreateOrUpdateTags](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將單一標籤新增至指定的 Auto Scaling 群組。標籤索引鍵為 'myTag'，標籤值為 'myTagValue'。Auto Scaling 會將此標籤傳播到 Auto Scaling 群組啟動的後續 EC2 執行個體。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Set-ASTag -Tag @( @{{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} })
```

範例 2：使用 PowerShell 版本 2 時，必須使用 New-Object 來建立 Tag 參數的標籤。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
$tag.PropagateAtLaunch = $true  
Set-ASTag -Tag $tag
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [CreateOrUpdateTags](#)。

Tools for PowerShell V5

範例 1：此範例將單一標籤新增至指定的 Auto Scaling 群組。標籤索引鍵為 'myTag'，標籤值為 'myTagValue'。Auto Scaling 會將此標籤傳播到 Auto Scaling 群組啟動的後續 EC2 執行個體。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Set-ASTag -Tag @( @{{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} })
```

範例 2：使用 PowerShell 版本 2 時，必須使用 New-Object 來建立 Tag 參數的標籤。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
$tag.PropagateAtLaunch = $true  
Set-ASTag -Tag $tag
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [CreateOrUpdateTags](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DeleteAutoScalingGroup 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DeleteAutoScalingGroup。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)
- [建置及管理彈性服務](#)

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

將 Auto Scaling 群組的最小大小更新為零，終止群組中的所有執行個體，然後刪除該群組。

```
/// <summary>
/// Try to terminate an instance by its Id.
/// </summary>
/// <param name="instanceId">The Id of the instance to terminate.</param>
/// <returns>Async task.</returns>
public async Task TryTerminateInstanceById(string instanceId)
{
    var stopping = false;
    Console.WriteLine($"Stopping {instanceId}...");
    while (!stopping)
    {
        try
```

```

        {
            await
            _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                new TerminateInstanceInAutoScalingGroupRequest()
                {
                    InstanceId = instanceId,
                    ShouldDecrementDesiredCapacity = false
                });
            stopping = true;
        }
        catch (ScalingActivityInProgressException)
        {
            Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
/// waits and retries until the group is successfully deleted.
/// </summary>
/// <param name="groupName">The name of the group to try to delete.</param>
/// <returns>Async task.</returns>
public async Task TryDeleteGroupByName(string groupName)
{
    var stopped = false;
    while (!stopped)
    {
        try
        {
            await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                new DeleteAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName
                });
            stopped = true;
        }
        catch (Exception e)
            when ((e is ScalingActivityInProgressException)
                || (e is Amazon.AutoScaling.Model.ResourceInUseException))
        {
    }
}

```

```

        Console.WriteLine($"Some instances are still running.
Waiting...");
        Thread.Sleep(10000);
    }
}

/// <summary>
/// Terminate instances and delete the Auto Scaling group by name.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>Async task.</returns>
public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
{
    var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
    new DescribeAutoScalingGroupsRequest()
    {
        AutoScalingGroupNames = new List<string>() { groupName }
    });
    if (describeGroupsResponse.AutoScalingGroups.Any())
    {
        // Update the size to 0.
        await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
            new UpdateAutoScalingGroupRequest()
            {
                AutoScalingGroupName = groupName,
                MinSize = 0
            });
        var group = describeGroupsResponse.AutoScalingGroups[0];
        foreach (var instance in group.Instances)
        {
            await TryTerminateInstanceById(instance.InstanceId);
        }

        await TryDeleteGroupByName(groupName);
    }
    else
    {
        Console.WriteLine($"No groups found with name {groupName}.");
    }
}

```

```
/// <summary>
/// Delete an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteAutoScalingGroupAsync(
    string groupName)
{
    var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        ForceDelete = true,
    };

    var response = await
        _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You successfully deleted {groupName}");
        return true;
    }

    Console.WriteLine($"Couldn't delete {groupName}.");
    return false;
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [DeleteAutoScalingGroup](#)。

C++

適用於 C++ 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);

Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
    autoScalingClient.DeleteAutoScalingGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
    result = false;
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DeleteAutoScalingGroup](#)。

CLI

AWS CLI

範例 1：刪除指定的 Auto Scaling 群組

此範例會刪除指定的 Auto Scaling 群組。

```
aws autoscaling delete-auto-scaling-group \  
  --auto-scaling-group-name my-asg
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [刪除 Auto Scaling 基礎架構](#)。

範例 2：強制刪除指定的 Auto Scaling 群組

若要刪除 Auto Scaling 群組，而不等待群組中的執行個體終止，請使用 `--force-delete` 選項。

```
aws autoscaling delete-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --force-delete
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [刪除 Auto Scaling 基礎架構](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteAutoScalingGroup](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName>

            Where:
                groupName - The name of the Auto Scaling group.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName) {
        try {
```



```

        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build();

autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DeleteAutoScalingGroup](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}

```

```
}  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DeleteAutoScalingGroup](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function deleteAutoScalingGroup($autoScalingGroupName)  
{  
    return $this->autoScalingClient->deleteAutoScalingGroup([  
        'AutoScalingGroupName' => $autoScalingGroupName,  
        'ForceDelete' => true,  
    ]);  
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [DeleteAutoScalingGroup](#)。

PowerShell

Tools for PowerShell V4

範例 1：如果沒有執行中的執行個體，此範例會刪除指定的 Auto Scaling 群組。在繼續操作之前，系統會提示您確認。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

範例 3：此範例會刪除指定的 Auto Scaling 群組，並終止其包含的任何執行中執行個體。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DeleteAutoScalingGroup](#)。

Tools for PowerShell V5

範例 1：如果沒有執行中的執行個體，此範例會刪除指定的 Auto Scaling 群組。在繼續操作之前，系統會提示您確認。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

範例 3：此範例會刪除指定的 Auto Scaling 群組，並終止其包含的任何執行中執行個體。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DeleteAutoScalingGroup](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

將 Auto Scaling 群組的最小大小更新為零，終止群組中的所有執行個體，然後刪除該群組。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
            created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
            that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
```

```

:param ec2_client: A Boto3 EC2 client.
:param ssm_client: A Boto3 Systems Manager client.
:param iam_client: A Boto3 IAM client.
"""

self.inst_type = inst_type
self.ami_param = ami_param
self.autoscaling_client = autoscaling_client
self.ec2_client = ec2_client
self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def delete_autoscaling_group(self, group_name: str) -> None:
    """
    Terminates all instances in the group, then deletes the EC2 Auto Scaling
    group.

    :param group_name: The name of the group to delete.
    """
    try:
        response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[group_name]
        )
        groups = response.get("AutoScalingGroups", [])
        if len(groups) > 0:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, MinSize=0
            )

```

```

        instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
        for inst_id in instance_ids:
            self.terminate_instance(inst_id)

        # Wait for all instances to be terminated
        if instance_ids:
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for all instances to be terminated...")
            waiter.wait(InstanceIds=instance_ids)
            log.info("All instances have been terminated.")
        else:
            log.info(f"No groups found named '{group_name}'! Nothing to do.")
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to delete the group again."
            )
        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the group."
            )
        log.error(f"Full error:\n\t{err}")

```

- 如需 API 詳細資訊，請參閱《AWS 適用於 Python (Boto3) 的 SDK API 參考》中的 [DeleteAutoScalingGroup](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(),
Error> {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
}
```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的 AWS SDK API 參考》中的 [DeleteAutoScalingGroup](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
" Example: iv_group_name = 'my-auto-scaling-group'
```

```

TRY.
  ao_asc->deleteautoscalinggroup(
    iv_autoscalinggroupname = iv_group_name ).

    " Wait for the group to be deleted (simplified - in production use proper
    polling)
    WAIT UP TO 10 SECONDS.

    MESSAGE 'Auto Scaling group deleted successfully' TYPE 'I'.

  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_ascresourceinusefault INTO DATA(lo_resource_in_use).
    RAISE EXCEPTION lo_resource_in_use.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DeleteAutoScalingGroup](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DeleteLaunchConfiguration** 與 CLI

下列程式碼範例示範如何使用 DeleteLaunchConfiguration。

CLI

AWS CLI

刪除啟動組態

此範例會刪除指定的啟動組態。

```
aws autoscaling delete-launch-configuration \
  --launch-configuration-name my-launch-config
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [刪除 Auto Scaling 基礎架構](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteLaunchConfiguration](#)。

PowerShell

Tools for PowerShell V4

範例 1：如果未連接至 Auto Scaling 群組，此範例會刪除指定的啟動組態。在繼續操作之前，系統會提示您確認。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
on Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DeleteLaunchConfiguration](#)。

Tools for PowerShell V5

範例 1：如果未連接至 Auto Scaling 群組，此範例會刪除指定的啟動組態。在繼續操作之前，系統會提示您確認。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

輸出：

```
Confirm
Are you sure you want to perform this action?
```

```
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
on Target "my-lc".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DeleteLaunchConfiguration](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DeleteLifecycleHook** 與 CLI

下列程式碼範例示範如何使用 DeleteLifecycleHook。

CLI

AWS CLI

刪除 lifecycle hook

此範例會刪除指定的 lifecycle hook。

```
aws autoscaling delete-lifecycle-hook \
  --lifecycle-hook-name my-lifecycle-hook \
  --auto-scaling-group-name my-asg
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteLifecycleHook](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例刪除指定 Auto Scaling 群組的指定 lifecycle hook。在繼續操作之前，系統會提示您確認。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook
```

輸出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target  
"myLifecycleHook".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DeleteLifecycleHook](#)。

Tools for PowerShell V5

範例 1：此範例刪除指定 Auto Scaling 群組的指定 lifecycle hook。在繼續操作之前，系統會提示您確認。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook
```

輸出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target  
"myLifecycleHook".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DeleteLifecycleHook](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DeleteNotificationConfiguration** 與 CLI

下列程式碼範例示範如何使用 DeleteNotificationConfiguration。

CLI

AWS CLI

刪除 Auto Scaling 通知

此範例從指定的 Auto Scaling 群組刪除指定的通知。

```
aws autoscaling delete-notification-configuration \  
  --auto-scaling-group-name my-asg \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [刪除通知組態](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteNotificationConfiguration](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例會刪除指定的通知動作。在繼續操作之前，系統會提示您確認。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN  
"arn:aws:sns:us-west-2:123456789012:my-topic"
```

輸出：

```
Confirm  
Are you sure you want to perform this action?
```

```
Performing operation "Remove-ASNotificationConfiguration
(DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
"arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DeleteNotificationConfiguration](#)。

Tools for PowerShell V5

範例 1：此範例會刪除指定的通知動作。在繼續操作之前，系統會提示您確認。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
"arn:aws:sns:us-west-2:123456789012:my-topic"
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
(DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
"arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DeleteNotificationConfiguration](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DeletePolicy** 與 CLI

下列程式碼範例示範如何使用 DeletePolicy。

CLI

AWS CLI

刪除擴展政策

此範例會刪除指定的擴展政策。

```
aws autoscaling delete-policy \  
  --auto-scaling-group-name my-asg \  
  --policy-name alb1000-target-tracking-scaling-policy
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeletePolicy](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例會刪除指定之 Auto Scaling 群組的指定政策。在繼續操作之前，系統會提示您確認。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

輸出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target  
"myScaleInPolicy".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DeletePolicy](#)。

Tools for PowerShell V5

範例 1：此範例會刪除指定之 Auto Scaling 群組的指定政策。在繼續操作之前，系統會提示您確認。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
"myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DeletePolicy](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DeleteScheduledAction** 與 CLI

下列程式碼範例示範如何使用 DeleteScheduledAction。

CLI

AWS CLI

刪除 Auto Scaling 群組的排程動作

此範例從指定的 Auto Scaling 群組刪除指定的排程動作。

```
aws autoscaling delete-scheduled-action \
```

```
--auto-scaling-group-name my-asg \  
--scheduled-action-name my-scheduled-action
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteScheduledAction](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例會刪除指定 Auto Scaling 群組的指定排程動作。在繼續操作之前，系統會提示您確認。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction"
```

輸出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target  
"myScheduledAction".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction" -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DeleteScheduledAction](#)。

Tools for PowerShell V5

範例 1：此範例會刪除指定 Auto Scaling 群組的指定排程動作。在繼續操作之前，系統會提示您確認。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction"
```


輸出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
"myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
"myScheduledAction" -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DeleteScheduledAction](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DeleteTags** 與 CLI

下列程式碼範例示範如何使用 DeleteTags。

CLI

AWS CLI

從 Auto Scaling 群組刪除標籤

此範例會從指定的 Auto Scaling 群組刪除指定的標籤。

```
aws autoscaling delete-tags \
  --tags ResourceId=my-asg, ResourceType=auto-scaling-
group, Key=Dept, Value=Research
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [標記 Auto Scaling 群組和執行個體](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteTags](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例從指定的 Auto Scaling 群組中移除指定的標籤。在繼續操作之前，系統會提示您確認。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Remove-ASTag -Tag @( @{{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
    Key="myTag" } } )
```

輸出：

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-ASTag (DeleteTags)" on target  
  "Amazon.AutoScaling.Model.Tag".  
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is  
  "Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASTag -Tag @( @{{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
    Key="myTag" } } ) -Force
```

範例 3：使用 Powershell 版本 2 時，必須使用 New-Object 來建立 Tag 參數的標籤。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
Remove-ASTag -Tag $tag -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DeleteTags](#)。

Tools for PowerShell V5

範例 1：此範例從指定的 Auto Scaling 群組中移除指定的標籤。在繼續操作之前，系統會提示您確認。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Remove-ASTag -Tag @( @{{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
    Key="myTag" } } )
```

輸出：

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-ASTag (DeleteTags)" on target  
  "Amazon.AutoScaling.Model.Tag".  
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is  
  "Y"):
```

範例 2：如果您指定強制參數，在繼續操作之前，系統不會提示您進行確認。

```
Remove-ASTag -Tag @( @{{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
    Key="myTag" } } ) -Force
```

範例 3：使用 Powershell 版本 2 時，必須使用 New-Object 來建立 Tag 參數的標籤。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
Remove-ASTag -Tag $tag -Force
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DeleteTags](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeAccountLimits** 與 CLI

下列程式碼範例示範如何使用 DescribeAccountLimits。

CLI

AWS CLI

描述您的 Amazon EC2 Auto Scaling 帳戶限制

此範例說明您 AWS 帳戶的 Amazon EC2 Auto Scaling 限制。

```
aws autoscaling describe-account-limits
```

輸出：

```
{
  "NumberOfLaunchConfigurations": 5,
  "MaxNumberOfLaunchConfigurations": 100,
  "NumberOfAutoScalingGroups": 3,
  "MaxNumberOfAutoScalingGroups": 20
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling Service Quotas](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeAccountLimits](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例說明您 AWS 帳戶的 Auto Scaling 資源限制。

```
Get-ASAccountLimit
```

輸出：

```
MaxNumberOfAutoScalingGroups      : 20
MaxNumberOfLaunchConfigurations    : 100
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeAccountLimits](#)。

Tools for PowerShell V5

範例 1：此範例說明您 AWS 帳戶的 Auto Scaling 資源限制。

```
Get-ASAccountLimit
```

輸出：

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeAccountLimits](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeAdjustmentTypes** 與 CLI

下列程式碼範例示範如何使用 `DescribeAdjustmentTypes`。

CLI

AWS CLI

描述可用的擴展調整類型

此範例描述可用的調整類型。

```
aws autoscaling describe-adjustment-types
```

輸出：

```
{
  "AdjustmentTypes": [
    {
      "AdjustmentType": "ChangeInCapacity"
    },
    {
      "AdjustmentType": "ExactCapacity"
    },
    {
      "AdjustmentType": "PercentChangeInCapacity"
    }
  ]
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [擴展調整類型](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeAdjustmentTypes](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例描述 Auto Scaling 支援的調整類型。

```
Get-ASAdjustmentType
```

輸出：

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeAdjustmentTypes](#)。

Tools for PowerShell V5

範例 1：此範例描述 Auto Scaling 支援的調整類型。

```
Get-ASAdjustmentType
```

輸出：

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeAdjustmentTypes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DescribeAutoScalingGroups 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DescribeAutoScalingGroups。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)
- [建置及管理彈性服務](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>
/// Get data about the instances in an Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
public async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    string groupName)
{
    var groups = await DescribeAutoScalingGroupsAsync(groupName);
    var instanceIds = new List<string>();
    var instanceDetails = new List<AutoScalingInstanceDetails>();
    if (groups != null)
    {
        groups.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
        });
    }
}
```

```
        }
    });

    var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
    {
        MaxRecords = 10,
        InstanceIds = instanceIds,
    };

    var response =
        await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
            scalingGroupsRequest);
    if (response.AutoScalingInstances != null)
    {
        instanceDetails = response.AutoScalingInstances;
    }
}

return instanceDetails;
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
```



```

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
    client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                << outcome.GetError().GetMessage()
                << std::endl;
}

```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

CLI

AWS CLI

範例 1：描述指定的 Auto Scaling 群組

此範例描述指定的 Auto Scaling 群組。

```
aws autoscaling describe-auto-scaling-groups \
    --auto-scaling-group-names my-asg
```

輸出：

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",

```

```

    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1",
      "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "MinSize": 0,
    "MaxSize": 1,
    "DesiredCapacity": 1,
    "DefaultCooldown": 300,
    "AvailabilityZones": [
      "us-west-2a",
      "us-west-2b",
      "us-west-2c"
    ],
    "LoadBalancerNames": [],
    "TargetGroupARNs": [],
    "HealthCheckType": "EC2",
    "HealthCheckGracePeriod": 0,
    "Instances": [
      {
        "InstanceId": "i-06905f55584de02da",
        "InstanceType": "t2.micro",
        "AvailabilityZone": "us-west-2a",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "ProtectedFromScaleIn": false,
        "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1",
          "LaunchTemplateId": "lt-1234567890abcde12"
        }
      }
    ],
    "CreatedTime": "2023-10-28T02:39:22.152Z",
    "SuspendedProcesses": [],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-
c934b782",
    "EnabledMetrics": [],
    "Tags": [],
    "TerminationPolicies": [
      "Default"
    ],
    "NewInstancesProtectedFromScaleIn": false,
    "ServiceLinkedRoleARN": "arn",

```

```
        "TrafficSources": []  
      }  
    ]  
  }  
}
```

範例 2：描述前 100 個指定的 Auto Scaling 群組

此範例描述指定的 Auto Scaling 群組。它可讓您指定最多 100 個群組名稱。

```
aws autoscaling describe-auto-scaling-groups \  
  --max-items 100 \  
  --auto-scaling-group-names "group1" "group2" "group3" "group4"
```

如需範例輸出，請參閱範例 1。

範例 3：描述指定區域中的 Auto Scaling 群組

此範例描述指定區域中的 Auto Scaling 群組，最多 75 個群組。

```
aws autoscaling describe-auto-scaling-groups \  
  --max-items 75 \  
  --region us-east-1
```

如需範例輸出，請參閱範例 1。

範例 4：描述 Auto Scaling 群組的指定數量

若要傳回特定數量的 Auto Scaling 群組，請使用 `--max-items` 選項。

```
aws autoscaling describe-auto-scaling-groups \  
  --max-items 1
```

如需範例輸出，請參閱範例 1。

如果輸出包含 `NextToken` 欄位，則有更多群組。若要取得其他群組，請在後續呼叫時使用此欄位的值和 `--starting-token` 選項，如下所示。

```
aws autoscaling describe-auto-scaling-groups \  
  --starting-token Z3M3LMPEXAMPLE
```

如需範例輸出，請參閱範例 1。

範例 5：描述使用啟動組態的 Auto Scaling 群組

此範例使用 `--query` 選項來描述使用啟動組態的 Auto Scaling 群組。

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

輸出：

```
[
  {
    "AutoScalingGroupName": "my-asg",
    "AutoScalingGroupARN": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480f03:autoScalingGroupName/my-asg",
    "LaunchConfigurationName": "my-lc",
    "MinSize": 0,
    "MaxSize": 1,
    "DesiredCapacity": 1,
    "DefaultCooldown": 300,
    "AvailabilityZones": [
      "us-west-2a",
      "us-west-2b",
      "us-west-2c"
    ],
    "LoadBalancerNames": [],
    "TargetGroupARNs": [],
    "HealthCheckType": "EC2",
    "HealthCheckGracePeriod": 0,
    "Instances": [
      {
        "InstanceId": "i-088c57934a6449037",
        "InstanceType": "t2.micro",
        "AvailabilityZone": "us-west-2c",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc",
        "ProtectedFromScaleIn": false
      }
    ],
    "CreatedTime": "2023-10-28T02:39:22.152Z",
    "SuspendedProcesses": [],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
```

```

        "EnabledMetrics": [],
        "Tags": [],
        "TerminationPolicies": [
            "Default"
        ],
        "NewInstancesProtectedFromScaleIn": false,
        "ServiceLinkedRoleARN": "arn",
        "TrafficSources": []
    }
]

```

如需詳細資訊，請參閱《AWS 命令列界面使用者指南》中的[篩選 AWS CLI 輸出](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[DescribeAutoScalingGroups](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 */

```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName>

            Where:
                groupName - The name of the Auto Scaling group.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient,
        String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
                DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
                    group.autoScalingGroupName());
            }
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

```

        System.out.println("The group ARN is " +
group.autoScalingGroupARN());

        List<Instance> instances = group.instances();
        for (Instance instance : instances) {
            instanceId = instance.instanceId();
        }
        return instanceId;
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DescribeAutoScalingGroups](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->

```

```
println("The group name is ${group.autoScalingGroupName}")
println("The group ARN is ${group.autoScalingGroupArn}")
group.instances?.forEach { instance ->
    println("The instance id is ${instance.instanceId}")
    println("The lifecycle state is " + instance.lifecycleState)
}
}
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function describeAutoScalingGroups($autoScalingGroupNames)
{
    return $this->autoScalingClient->describeAutoScalingGroups([
        'AutoScalingGroupNames' => $autoScalingGroupNames
    ]);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例列出 Auto Scaling 群組的名稱。


```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

輸出：

```
AutoScalingGroupName
-----
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

範例 2：此範例描述指定的 Auto Scaling 群組。

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

輸出：

```
AutoScalingGroupARN      : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                           f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName     : my-asg-1
AvailabilityZones        : {us-west-2b, us-west-2a}
CreatedTime              : 3/1/2015 9:05:31 AM
DefaultCooldown          : 300
DesiredCapacity          : 2
EnabledMetrics           : {}
HealthCheckGracePeriod   : 300
HealthCheckType          : EC2
Instances                : {my-1c}
LaunchConfigurationName  : my-1c
LoadBalancerNames        : {}
MaxSize                  : 0
MinSize                  : 0
PlacementGroup           :
Status                   :
SuspendedProcesses       : {}
Tags                     : {}
TerminationPolicies      : {Default}
VPCZoneIdentifier        : subnet-e4f33493,subnet-5264e837
```

範例 3：此範例描述指定的兩個 Auto Scaling 群組。

```
Get-ASAutoScalingGroup -AutoScalingGroupName @"my-asg-1", "my-asg-2")
```

範例 4：此範例描述指定之 Auto Scaling 群組的 Auto Scaling 執行個體。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

範例 5：此範例描述所有 Auto Scaling 群組。

```
Get-ASAutoScalingGroup
```

範例 6：此範例描述指定的 Auto Scaling 群組的 LaunchTemplate。此範例假設「執行個體購買選項」設定為「依循啟動範本」。如果此選項設定為「結合購買選項和執行個體類型」，則可使用 "MixedInstancesPolicy.LaunchTemplate" 屬性存取 LaunchTemplate。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

輸出：

LaunchTemplateId	LaunchTemplateName	Version
lt-06095fd619cb40371	test-launch-template	\$Default

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeAutoScalingGroups](#)。

Tools for PowerShell V5

範例 1：此範例列出 Auto Scaling 群組的名稱。

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

輸出：

```
AutoScalingGroupName
-----
my-asg-1
my-asg-2
```

```
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

範例 2：此範例描述指定的 Auto Scaling 群組。

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

輸出：

```
AutoScalingGroupARN      : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                           f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName     : my-asg-1
AvailabilityZones        : {us-west-2b, us-west-2a}
CreatedTime              : 3/1/2015 9:05:31 AM
DefaultCooldown          : 300
DesiredCapacity          : 2
EnabledMetrics           : {}
HealthCheckGracePeriod   : 300
HealthCheckType          : EC2
Instances                : {my-1c}
LaunchConfigurationName  : my-1c
LoadBalancerNames        : {}
MaxSize                  : 0
MinSize                  : 0
PlacementGroup           :
Status                   :
SuspendedProcesses       : {}
Tags                    : {}
TerminationPolicies      : {Default}
VPCZoneIdentifier        : subnet-e4f33493,subnet-5264e837
```

範例 3：此範例描述指定的兩個 Auto Scaling 群組。

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

範例 4：此範例描述指定之 Auto Scaling 群組的 Auto Scaling 執行個體。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

範例 5：此範例描述所有 Auto Scaling 群組。

```
Get-ASAutoScalingGroup
```

範例 6：此範例描述指定的 Auto Scaling 群組的 LaunchTemplate。此範例假設「執行個體購買選項」設定為「依循啟動範本」。如果此選項設定為「結合購買選項和執行個體類型」，則可使用 "MixedInstancesPolicy.LaunchTemplate" 屬性存取 LaunchTemplate。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

輸出：

LaunchTemplateId	LaunchTemplateName	Version
lt-06095fd619cb40371	test-launch-template	\$Default

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeAutoScalingGroups](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client
```

```

def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
    """
    Gets information about an Auto Scaling group.

    :param group_name: The name of the group to look up.
    :return: A dictionary with information about the group if found,
    otherwise None.
    :raises ClientError: If there is an error describing the Auto Scaling
    group.
    """
    try:
        paginator = self.autoscaling_client.get_paginator(
            "describe_auto_scaling_groups"
        )
        response_iterator =
paginator.paginate(AutoScalingGroupNames=[group_name])
        groups = []
        for response in response_iterator:
            groups.extend(response.get("AutoScalingGroups", []))

        logger.info(
            f"Successfully retrieved information for Auto Scaling group
{group_name}."
        )

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(f"Failed to describe Auto Scaling group {group_name}.")
        if error_code == "ResourceContentionFault":
            logger.error(
                "There is a conflict with another operation that is modifying
the "
                f"Auto Scaling group '{group_name}' Please try again later."
            )
        logger.error(f"Full error:\n\t{err}")
        raise
    else:
        return groups[0] if len(groups) > 0 else None

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [DescribeAutoScalingGroups](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name: {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn: {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [DescribeAutoScalingGroups](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

DATA lt_group_names TYPE /aws1/
cl_ascautoscgroupnames_w=>tt_autoscalinggroupnames.
DATA lo_group_name TYPE REF TO /aws1/cl_ascautoscgroupnames_w.

" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
  " Build group names parameter
  CREATE OBJECT lo_group_name
    EXPORTING
      iv_value = iv_group_name.
  APPEND lo_group_name TO lt_group_names.

  " Describe the Auto Scaling group
  DATA(lo_output) = ao_asc->describeautoscalinggroups(
    it_autoscalinggroupnames = lt_group_names ).

  " Return the first (and only) group in the result
  DATA(lt_groups) = lo_output->get_autoscalinggroups( ).
  IF lines( lt_groups ) > 0.
    READ TABLE lt_groups INDEX 1 INTO DATA(lo_group).
    oo_output = lo_group.
  ENDIF.

  MESSAGE 'Auto Scaling group information retrieved successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DescribeAutoScalingGroups](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DescribeAutoScalingInstances 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DescribeAutoScalingInstances。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>
/// Get data about the instances in an Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
public async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    string groupName)
{
    var groups = await DescribeAutoScalingGroupsAsync(groupName);
    var instanceIds = new List<string>();
    var instanceDetails = new List<AutoScalingInstanceDetails>();
    if (groups != null)
```



```
{
    groups.ForEach(group =>
    {
        if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
        {
            group.Instances.ForEach(instance =>
            {
                instanceIds.Add(instance.InstanceId);
            });
        }
    });

    var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
    {
        MaxRecords = 10,
        InstanceIds = instanceIds,
    };

    var response =
        await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
            scalingGroupsRequest);
    if (response.AutoScalingInstances != null)
    {
        instanceDetails = response.AutoScalingInstances;
    }
}

return instanceDetails;
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [DescribeAutoScalingInstances](#)。

C++

適用於 C++ 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
request.SetInstanceIds(instanceIDs);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
    client.DescribeAutoScalingInstances(request);

if (outcome.IsSuccess()) {
    const
    Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
    &instancesDetails =
        outcome.GetResult().GetAutoScalingInstances();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                << outcome.GetError().GetMessage()
                << std::endl;
    return false;
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DescribeAutoScalingInstances](#)。

CLI

AWS CLI

範例 1：描述一或多個執行個體

此範例描述指定的執行個體。

```
aws autoscaling describe-auto-scaling-instances \  
  --instance-ids i-06905f55584de02da
```

輸出：

```
{  
  "AutoScalingInstances": [  
    {  
      "InstanceId": "i-06905f55584de02da",  
      "InstanceType": "t2.micro",  
      "AutoScalingGroupName": "my-asg",  
      "AvailabilityZone": "us-west-2b",  
      "LifecycleState": "InService",  
      "HealthStatus": "HEALTHY",  
      "ProtectedFromScaleIn": false,  
      "LaunchTemplate": {  
        "LaunchTemplateId": "lt-1234567890abcde12",  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1"  
      }  
    }  
  ]  
}
```

範例 2：描述一或多個執行個體

此範例使用 `--max-items` 選項指定此呼叫傳回的執行個體數量。

```
aws autoscaling describe-auto-scaling-instances \  
  --max-items 1
```

如果輸出包含 `NextToken` 欄位，則有更多執行個體。若要取得其他執行個體，請在後續呼叫中使用此欄位的值和 `--starting-token` 選項，如下所示。

```
aws autoscaling describe-auto-scaling-instances \
  --starting-token Z3M3LMPEXAMPLE
```

如需範例輸出，請參閱範例 1。

範例 3：描述使用啟動組態的執行個體

此範例使用 `--query` 選項來描述使用啟動組態的執行個體。

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?LaunchConfigurationName!=`null`]'
```

輸出：

```
[
  {
    "InstanceId": "i-088c57934a6449037",
    "InstanceType": "t2.micro",
    "AutoScalingGroupName": "my-asg",
    "AvailabilityZone": "us-west-2c",
    "LifecycleState": "InService",
    "HealthStatus": "HEALTHY",
    "LaunchConfigurationName": "my-lc",
    "ProtectedFromScaleIn": false
  }
]
```

如需詳細資訊，請參閱《AWS 命令列界面使用者指南》中的[篩選 AWS CLI 輸出](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[DescribeAutoScalingInstances](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DescribeAutoScalingInstances](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
```

```

        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

        AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
            val response =
            autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
            response.autoScalingInstances?.forEach { group ->
                println("The instance lifecycle state is: ${group.lifecycleState}")
            }
        }
    }
}

```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DescribeAutoScalingInstances](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

public function describeAutoScalingInstances($instanceIds)
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
}

```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [DescribeAutoScalingInstances](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例列出 Auto Scaling 執行個體的 ID。

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

輸出：

```
InstanceId
-----
i-12345678
i-87654321
i-abcd1234
```

範例 2：此範例描述指定的 Auto Scaling 執行個體。

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

輸出：

```
AutoScalingGroupName    : my-asg
AvailabilityZone         : us-west-2b
HealthStatus             : HEALTHY
InstanceId               : i-12345678
LaunchConfigurationName : my-lc
LifecycleState           : InService
```

範例 3：此範例描述指定的兩個 Auto Scaling 執行個體。

```
Get-ASAutoScalingInstance -InstanceId @"i-12345678", "i-87654321"
```

範例 4：此範例描述指定之 Auto Scaling 群組的 Auto Scaling 執行個體。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-ASAutoScalingInstance
```

範例 5：此範例描述所有 Auto Scaling 執行個體。

```
Get-ASAutoScalingInstance
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeAutoScalingInstances](#)。

Tools for PowerShell V5

範例 1：此範例列出 Auto Scaling 執行個體的 ID。

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

輸出：

```
InstanceId
-----
i-12345678
i-87654321
i-abcd1234
```

範例 2：此範例描述指定的 Auto Scaling 執行個體。

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

輸出：

```
AutoScalingGroupName    : my-asg
AvailabilityZone         : us-west-2b
HealthStatus            : HEALTHY
InstanceId               : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

範例 3：此範例描述指定的兩個 Auto Scaling 執行個體。

```
Get-ASAutoScalingInstance -InstanceId @( "i-12345678", "i-87654321" )
```

範例 4：此範例描述指定之 Auto Scaling 群組的 Auto Scaling 執行個體。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-ASAutoScalingInstance
```


範例 5：此範例描述所有 Auto Scaling 執行個體。

Get-ASAutoScalingInstance

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeAutoScalingInstances](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def describe_instances(self, instance_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: A list of dictionaries with information about each instance,
            or an empty list if none are found.
        :raises ClientError: If there is an error describing the instances.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_instances"
            )
```

```

        response_iterator = paginator.paginate(InstanceIds=instance_ids)

        instances = []
        for response in response_iterator:
            instances.extend(response.get("AutoScalingInstances", []))

        logger.info(f"Successfully described instances: {instance_ids}")

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(
            f"Couldn't describe instances {instance_ids}. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
        )
        raise
    else:
        return instances

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [DescribeAutoScalingInstances](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
    // The direct way to list instances is by using
    DescribeAutoScalingGroup's instances property. However, this returns a
    Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
    // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
    i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
    id.is_empty()).collect())

```

```
// Alternatively, and for the sake of example,
DescribeAutoScalingInstances returns a list that can be filtered by the client.
self.autoscaling
    .describe_auto_scaling_instances()
    .into_paginator()
    .items()
    .send()
    .try_collect()
    .await
    .map(|items| {
        items
            .into_iter()
            .filter(|i| {
                i.auto_scaling_group_name.as_deref()
                    == Some(self.auto_scaling_group_name.as_str())
            })
            .map(|i| i.instance_id.unwrap_or_default())
            .filter(|id| !id.is_empty())
            .collect::<Vec<String>>()
    })
    .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [DescribeAutoScalingInstances](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

" Example: it_instance_ids contains a list of instance IDs

TRY.

```

DATA(lo_output) = ao_asc->describeautoscalinginstances(
    it_instanceids = it_instance_ids ).

ot_output = lo_output->get_autoscalinginstances( ).

MESSAGE 'Auto Scaling instances information retrieved successfully' TYPE
'I'.

CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DescribeAutoScalingInstances](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeAutoScalingNotificationTypes** 與 CLI

下列程式碼範例示範如何使用 DescribeAutoScalingNotificationTypes。

CLI

AWS CLI

描述可用的通知類型

此範例描述可用的通知類型。

```
aws autoscaling describe-auto-scaling-notification-types
```

輸出：

```

{
  "AutoScalingNotificationTypes": [
    "autoscaling:EC2_INSTANCE_LAUNCH",
    "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
    "autoscaling:EC2_INSTANCE_TERMINATE",

```

```
        "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",  
        "autoscaling:TEST_NOTIFICATION"  
    ]  
}
```

有關詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的[取得 Auto Scaling 群組擴展時的 Amazon SNS 通知](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[DescribeAutoScalingNotificationTypes](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例列出 Auto Scaling 支援的通知類型。

```
Get-ASAutoScalingNotificationType
```

輸出：

```
autoscaling:EC2_INSTANCE_LAUNCH  
autoscaling:EC2_INSTANCE_LAUNCH_ERROR  
autoscaling:EC2_INSTANCE_TERMINATE  
autoscaling:EC2_INSTANCE_TERMINATE_ERROR  
autoscaling:TEST_NOTIFICATION
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的[DescribeAutoScalingNotificationTypes](#)。

Tools for PowerShell V5

範例 1：此範例列出 Auto Scaling 支援的通知類型。

```
Get-ASAutoScalingNotificationType
```

輸出：

```
autoscaling:EC2_INSTANCE_LAUNCH  
autoscaling:EC2_INSTANCE_LAUNCH_ERROR  
autoscaling:EC2_INSTANCE_TERMINATE
```

```
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeAutoScalingNotificationTypes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeLaunchConfigurations** 與 CLI

下列程式碼範例示範如何使用 DescribeLaunchConfigurations。

CLI

AWS CLI

範例 1：描述指定的啟動組態

此範例描述指定的啟動組態。

```
aws autoscaling describe-launch-configurations \
  --launch-configuration-names my-launch-config
```

輸出：

```
{
  "LaunchConfigurations": [
    {
      "LaunchConfigurationName": "my-launch-config",
      "LaunchConfigurationARN": "arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:98d3b196-4cf9-4e88-8ca1-8547c24ced8b:launchConfig
my-launch-config",
      "ImageId": "ami-0528a5175983e7f28",
      "KeyName": "my-key-pair-uswest2",
      "SecurityGroups": [
        "sg-05eaec502fcdadc2e"
      ],
      "ClassicLinkVPCSecurityGroups": [],
      "UserData": "",
      "InstanceType": "t2.micro",
      "KernelId": "",
```

```

        "RamdiskId": "",
        "BlockDeviceMappings": [
            {
                "DeviceName": "/dev/xvda",
                "Ebs": {
                    "SnapshotId": "snap-06c1606ba5ca274b1",
                    "VolumeSize": 8,
                    "VolumeType": "gp2",
                    "DeleteOnTermination": true,
                    "Encrypted": false
                }
            }
        ],
        "InstanceMonitoring": {
            "Enabled": true
        },
        "CreatedTime": "2020-10-28T02:39:22.321Z",
        "EbsOptimized": false,
        "AssociatePublicIpAddress": true,
        "MetadataOptions": {
            "HttpTokens": "required",
            "HttpPutResponseHopLimit": 1,
            "HttpEndpoint": "disabled"
        }
    }
]
}

```

範例 2：描述指定數量的啟動組態

若要傳回特定數量的啟動組態，請使用 `--max-items` 選項。

```
aws autoscaling describe-launch-configurations \
    --max-items 1
```

如果輸出包含 `NextToken` 欄位，則有更多啟動組態。若要取得其他啟動組態，請在後續呼叫中使用此欄位的值與 `--starting-token` 選項，如下所示。

```
aws autoscaling describe-launch-configurations \
    --starting-token Z3M3LMPEXAMPLE
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeLaunchConfigurations](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例列出啟動組態的名稱。

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

輸出：

```
LaunchConfigurationName
-----
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

範例 2：此範例描述指定的啟動組態。

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

輸出：

```
AssociatePublicIpAddress      : True
BlockDeviceMappings           : {/dev/xvda}
ClassicLinkVPCId              :
ClassicLinkVPCSecurityGroups  : {}
CreatedTime                   : 12/12/2014 3:22:08 PM
EbsOptimized                  : False
IamInstanceProfile            :
ImageId                       : ami-043a5034
InstanceMonitoring            : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                  : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN        : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName       : my-lc-1
PlacementTenancy              :
RamdiskId                    :
```



```
SecurityGroups      : {sg-67ef0308}
SpotPrice           :
UserData           :
```

範例 3：此範例描述指定的兩個啟動組態。

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

範例 4：此範例描述所有啟動組態。

```
Get-ASLaunchConfiguration
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeLaunchConfigurations](#)。

Tools for PowerShell V5

範例 1：此範例列出啟動組態的名稱。

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

輸出：

```
LaunchConfigurationName
-----
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

範例 2：此範例描述指定的啟動組態。

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

輸出：

```
AssociatePublicIpAddress : True
BlockDeviceMappings       : {/dev/xvda}
ClassicLinkVPCId          :
ClassicLinkVPCSecurityGroups : {}
```

```

CreatedTime           : 12/12/2014 3:22:08 PM
EbsOptimized          : False
IamInstanceProfile    :
ImageId               : ami-043a5034
InstanceMonitoring    : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType          : t2.micro
KernelId              :
KeyName               :
LaunchConfigurationARN : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName : my-lc-1
PlacementTenancy      :
RamdiskId             :
SecurityGroups        : {sg-67ef0308}
SpotPrice              :
UserData              :

```

範例 3：此範例描述指定的兩個啟動組態。

```
Get-ASLaunchConfiguration -LaunchConfigurationName @"("my-lc-1", "my-lc-2")
```

範例 4：此範例描述所有啟動組態。

```
Get-ASLaunchConfiguration
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeLaunchConfigurations](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeLifecycleHookTypes** 與 CLI

下列程式碼範例示範如何使用 `DescribeLifecycleHookTypes`。

CLI

AWS CLI

刪除可用的 lifecycle hook 類型。

此範例描述可用的 lifecycle hook 類型。

```
aws autoscaling describe-lifecycle-hook-types
```

輸出：

```
{
  "LifecycleHookTypes": [
    "autoscaling:EC2_INSTANCE_LAUNCHING",
    "autoscaling:EC2_INSTANCE_TERMINATING"
  ]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeLifecycleHookTypes](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例列出 Auto Scaling 支援的 lifecycle hook 類型。

```
Get-ASLifecycleHookType
```

輸出：

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeLifecycleHookTypes](#)。

Tools for PowerShell V5

範例 1：此範例列出 Auto Scaling 支援的 lifecycle hook 類型。

```
Get-ASLifecycleHookType
```

輸出：

```
autoscaling:EC2_INSTANCE_LAUNCHING
```

```
auto-scaling:EC2_INSTANCE_TERMINATING
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeLifecycleHookTypes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeLifecycleHooks** 與 CLI

下列程式碼範例示範如何使用 DescribeLifecycleHooks。

CLI

AWS CLI

描述您的 lifecycle hook

此範例描述指定之 Auto Scaling 群組的 lifecycle hook。

```
aws autoscaling describe-lifecycle-hooks \  
  --auto-scaling-group-name my-asg
```

輸出：

```
{  
  "LifecycleHooks": [  
    {  
      "GlobalTimeout": 3000,  
      "HeartbeatTimeout": 30,  
      "AutoScalingGroupName": "my-asg",  
      "LifecycleHookName": "my-launch-hook",  
      "DefaultResult": "ABANDON",  
      "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING"  
    },  
    {  
      "GlobalTimeout": 6000,  
      "HeartbeatTimeout": 60,  
      "AutoScalingGroupName": "my-asg",  
      "LifecycleHookName": "my-termination-hook",  
      "DefaultResult": "CONTINUE",  
      "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING"  
    }  
  ]  
}
```

```
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeLifecycleHooks](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例描述指定的 lifecycle hook。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
myLifecycleHook
```

輸出：

```
AutoScalingGroupName : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition    : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

範例 2：此範例描述指定的 Auto Scaling 群組的 lifecycle hook。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

範例 3：此範例描述所有 Auto Scaling 群組的所有 lifecycle hook。

```
Get-ASLifecycleHook
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeLifecycleHooks](#)。

Tools for PowerShell V5

範例 1：此範例描述指定的 lifecycle hook。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
myLifecycleHook
```

輸出：

```
AutoScalingGroupName   : my-asg
DefaultResult           : ABANDON
GlobalTimeout           : 172800
HeartbeatTimeout        : 3600
LifecycleHookName       : myLifecycleHook
LifecycleTransition      : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata    :
NotificationTargetARN    : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN                  : arn:aws:iam::123456789012:role/my-iam-role
```

範例 2：此範例描述指定的 Auto Scaling 群組的 lifecycle hook。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

範例 3：此範例描述所有 Auto Scaling 群組的所有 lifecycle hook。

```
Get-ASLifecycleHook
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeLifecycleHooks](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeLoadBalancers** 與 CLI

下列程式碼範例示範如何使用 DescribeLoadBalancers。

CLI

AWS CLI

描述 Auto Scaling 群組的 Classic Load Balancer

此範例描述指定之 Auto Scaling 群組的 Classic Load Balancer。

```
aws autoscaling describe-load-balancers \
  --auto-scaling-group-name my-asg
```

輸出：

```
{
  "LoadBalancers": [
    {
      "State": "Added",
      "LoadBalancerName": "my-load-balancer"
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeLoadBalancers](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例描述指定的 Auto Scaling 群組的負載平衡器。

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

輸出：

LoadBalancerName	State
-----	-----
my-lb	Added

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeLoadBalancers](#)。

Tools for PowerShell V5

範例 1：此範例描述指定的 Auto Scaling 群組的負載平衡器。

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

輸出：

LoadBalancerName	State
-----	-----
my-lb	Added

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeLoadBalancers](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeMetricCollectionTypes** 與 CLI

下列程式碼範例示範如何使用 `DescribeMetricCollectionTypes`。

CLI

AWS CLI

描述可用的指標收集類型

此範例描述可用的指標收集類型。

```
aws autoscaling describe-metric-collection-types
```

輸出：

```
{
  "Metrics": [
    {
      "Metric": "GroupMinSize"
    },
    {
      "Metric": "GroupMaxSize"
    },
    {
      "Metric": "GroupDesiredCapacity"
    },
    {
      "Metric": "GroupInServiceInstances"
    },
    {
      "Metric": "GroupInServiceCapacity"
    }
  ]
}
```



```

    },
    {
      "Metric": "GroupPendingInstances"
    },
    {
      "Metric": "GroupPendingCapacity"
    },
    {
      "Metric": "GroupTerminatingInstances"
    },
    {
      "Metric": "GroupTerminatingCapacity"
    },
    {
      "Metric": "GroupStandbyInstances"
    },
    {
      "Metric": "GroupStandbyCapacity"
    },
    {
      "Metric": "GroupTotalInstances"
    },
    {
      "Metric": "GroupTotalCapacity"
    }
  ],
  "Granularities": [
    {
      "Granularity": "1Minute"
    }
  ]
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Auto Scaling 群組指標](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeMetricCollectionTypes](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例列出 Auto Scaling 支援的指標收集類型。

```
(Get-ASMetricCollectionType).Metrics
```

輸出：

```
Metric
-----
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

範例 2：此範例列出對應的精細程度。

```
(Get-ASMetricCollectionType).Granularities
```

輸出：

```
Granularity
-----
1Minute
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeMetricCollectionTypes](#)。

Tools for PowerShell V5

範例 1：此範例列出 Auto Scaling 支援的指標收集類型。

```
(Get-ASMetricCollectionType).Metrics
```

輸出：

```
Metric
-----
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
```

```
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

範例 2：此範例列出對應的精細程度。

```
(Get-ASMetricCollectionType).Granularities
```

輸出：

```
Granularity
-----
1Minute
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeMetricCollectionTypes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeNotificationConfigurations** 與 CLI

下列程式碼範例示範如何使用 `DescribeNotificationConfigurations`。

CLI

AWS CLI

範例 1：描述指定群組的通知組態

此範例描述指定的 Auto Scaling 群組的通知組態。

```
aws autoscaling describe-notification-configurations \
  --auto-scaling-group-name my-asg
```

輸出：

```
{
  "NotificationConfigurations": [
    {
```

```

        "AutoScalingGroupName": "my-asg",
        "NotificationType": "autoscaling:TEST_NOTIFICATION",
        "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
    },
    {
        "AutoScalingGroupName": "my-asg",
        "NotificationType": "autoscaling:TEST_NOTIFICATION",
        "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
    }
]
}

```

有關詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的[取得 Auto Scaling 群組擴展時的 Amazon SNS 通知](#)。

範例 1：描述指定的通知組態數量

若要傳回特定數量的通知組態，請使用 `max-items` 參數。

```

aws autoscaling describe-notification-configurations \
  --auto-scaling-group-name my-auto-scaling-group \
  --max-items 1

```

輸出：

```

{
  "NotificationConfigurations": [
    {
      "AutoScalingGroupName": "my-asg",
      "NotificationType": "autoscaling:TEST_NOTIFICATION",
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
    },
    {
      "AutoScalingGroupName": "my-asg",
      "NotificationType": "autoscaling:TEST_NOTIFICATION",
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
    }
  ]
}

```

如果輸出包含 `NextToken` 欄位，則會有更多通知組態。若要取得其他通知組態，請在後續呼叫中搭配使用此欄位的值和 `starting-token` 參數，如下所示。

```
aws autoscaling describe-notification-configurations \  
  --auto-scaling-group-name my-asg \  
  --starting-token Z3M3LMPEXAMPLE
```

有關詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的[取得 Auto Scaling 群組擴展時的 Amazon SNS 通知](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[DescribeNotificationConfigurations](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例描述與指定的 Auto Scaling 群組相關聯的通知動作。

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

輸出：

```
AutoScalingGroupName : my-asg  
NotificationType      : auto-scaling:EC2_INSTANCE_LAUNCH  
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic  
  
AutoScalingGroupName : my-asg  
NotificationType      : auto-scaling:EC2_INSTANCE_TERMINATE  
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic
```

範例 2：此範例描述與所有 Auto Scaling 群組相關聯的通知動作。

```
Get-ASNotificationConfiguration
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的[DescribeNotificationConfigurations](#)。

Tools for PowerShell V5

範例 1：此範例描述與指定的 Auto Scaling 群組相關聯的通知動作。

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

輸出：

```
AutoScalingGroupName : my-asg
NotificationType      : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType      : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic
```

範例 2：此範例描述與所有 Auto Scaling 群組相關聯的通知動作。

```
Get-ASNotificationConfiguration
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeNotificationConfigurations](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribePolicies** 與 CLI

下列程式碼範例示範如何使用 DescribePolicies。

CLI

AWS CLI

範例 1：描述指定群組的擴展政策

此範例描述指定的 Auto Scaling 群組的擴展政策。

```
aws autoscaling describe-policies \
  --auto-scaling-group-name my-asg
```

輸出：

```
{
  "ScalingPolicies": [
    {
      "AutoScalingGroupName": "my-asg",
```

```

        "PolicyName": "alb1000-target-tracking-scaling-policy",
        "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:3065d9c8-9969-4bec-
bb6a-3fbe5550fde6:autoScalingGroupName/my-asg:policyName/alb1000-target-tracking-
scaling-policy",
        "PolicyType": "TargetTrackingScaling",
        "StepAdjustments": [],
        "Alarms": [
            {
                "AlarmName": "TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196",
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196"
            },
            {
                "AlarmName": "TargetTracking-my-asg-AlarmLow-f96f899d-
b8e7-4d09-a010-c1aaa35da296",
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-a010-
c1aaa35da296"
            }
        ],
        "TargetTrackingConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "ALBRequestCountPerTarget",
                "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-
alb-target-group/943f017f100becff"
            },
            "TargetValue": 1000.0,
            "DisableScaleIn": false
        },
        "Enabled": true
    },
    {
        "AutoScalingGroupName": "my-asg",
        "PolicyName": "cpu40-target-tracking-scaling-policy",
        "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:5fd26f71-39d4-4690-82a9-
b8515c45cdde:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-
scaling-policy",
        "PolicyType": "TargetTrackingScaling",
        "StepAdjustments": [],
        "Alarms": [

```

```

        {
            "AlarmName": "TargetTracking-my-asg-
AlarmHigh-139f9789-37b9-42ad-bea5-b5b147d7f473",
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-139f9789-37b9-42ad-
bea5-b5b147d7f473"
        },
        {
            "AlarmName": "TargetTracking-my-asg-AlarmLow-bd681c67-
fc18-4c56-8468-fb8e413009c9",
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-bd681c67-fc18-4c56-8468-
fb8e413009c9"
        }
    ],
    "TargetTrackingConfiguration": {
        "PredefinedMetricSpecification": {
            "PredefinedMetricType": "ASGAverageCPUUtilization"
        },
        "TargetValue": 40.0,
        "DisableScaleIn": false
    },
    "Enabled": true
}
]
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[動態擴展](#)。

範例 2：描述指定名稱的擴展政策

若要傳回特定擴展政策，請使用 `--policy-names` 選項。

```

aws autoscaling describe-policies \
  --auto-scaling-group-name my-asg \
  --policy-names cpu40-target-tracking-scaling-policy

```

如需範例輸出，請參閱範例 1。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[動態擴展](#)。

範例 3：描述一些擴展政策

若要傳回特定數量的政策，請使用 `--max-items` 選項。


```
aws autoscaling describe-policies \
  --auto-scaling-group-name my-asg \
  --max-items 1
```

如需範例輸出，請參閱範例 1。

如果輸出包含 NextToken 欄位，請在後續呼叫中搭配使用此欄位的值和 --starting-token 選項，以取得其他政策。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg --starting-token Z3M3LMPEXAMPLE
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[動態擴展](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribePolicies](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例描述指定的 Auto Scaling 群組的所有政策。

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

輸出：

```
AdjustmentType      : ChangeInCapacity
Alarms              : {}
AutoScalingGroupName : my-asg
Cooldown            : 0
EstimatedInstanceWarmup : 0
MetricAggregationType :
MinAdjustmentMagnitude : 0
MinAdjustmentStep    : 0
PolicyARN            : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                    : autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName           : myScaleInPolicy
PolicyType           : SimpleScaling
ScalingAdjustment     : -1
StepAdjustments       : {}
```

範例 2：此範例描述指定的 Auto Scaling 群組的指定政策。

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
"myScaleInPolicy")
```

範例 3：此範例描述所有 Auto Scaling 群組的所有政策。

```
Get-ASPolicy
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribePolicies](#)。

Tools for PowerShell V5

範例 1：此範例描述指定的 Auto Scaling 群組的所有政策。

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

輸出：

```
AdjustmentType      : ChangeInCapacity
Alarms              : {}
AutoScalingGroupName : my-asg
Cooldown            : 0
EstimatedInstanceWarmup : 0
MetricAggregationType :
MinAdjustmentMagnitude : 0
MinAdjustmentStep    : 0
PolicyARN            : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                    :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName           : myScaleInPolicy
PolicyType           : SimpleScaling
ScalingAdjustment    : -1
StepAdjustments       : {}
```

範例 2：此範例描述指定的 Auto Scaling 群組的指定政策。

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
"myScaleInPolicy")
```

範例 3：此範例描述所有 Auto Scaling 群組的所有政策。

Get-ASPolicy

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribePolicies](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DescribeScalingActivities 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DescribeScalingActivities。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>
/// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
/// Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
/// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
public async Task<List<Activity>> DescribeScalingActivitiesAsync(
    string groupName)
{
    var activities = new List<Activity>();
    var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
```

```
{
    AutoScalingGroupName = groupName,
    MaxRecords = 10,
};

var response = await
_amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
if (response.Activities != null)
{
    activities = response.Activities;
}
return activities;
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [DescribeScalingActivities](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
request.SetAutoScalingGroupName(groupName);

Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
Aws::String nextToken; // Used for pagination;
do {
```

```

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
            autoScalingClient.DescribeScalingActivities(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                outcome.GetResult().GetActivities();
            allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeScalingActivities."
"
                << outcome.GetError().GetMessage()
                << std::endl;

        }
    } while (!nextToken.empty());

    std::cout << "Found " << allActivities.size() << " activities."
        << std::endl;
    std::cout << "Activities are ordered with the most recent first."
        << std::endl;
    for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
        std::cout << activity.GetDescription() << std::endl;
        std::cout << activity.GetDetails() << std::endl;
    }
}

```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DescribeScalingActivities](#)。

CLI

AWS CLI

範例 1：描述指定群組的擴展活動

此範例描述指定的 Auto Scaling 群組的擴展活動。

```
aws autoscaling describe-scaling-activities \  
  --auto-scaling-group-name my-asg
```

輸出：

```
{  
  "Activities": [  
    {  
      "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",  
      "Description": "Launching a new EC2 instance: i-0d44425630326060f",  
      "AutoScalingGroupName": "my-asg",  
      "Cause": "At 2020-10-30T19:35:51Z a user request update of  
AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the  
desired capacity from 0 to 16. At 2020-10-30T19:36:07Z an instance was started  
in response to a difference between desired and actual capacity, increasing the  
capacity from 0 to 16.",  
      "StartTime": "2020-10-30T19:36:09.766Z",  
      "EndTime": "2020-10-30T19:36:41Z",  
      "StatusCode": "Successful",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":  
\"us-west-2b\"}"  
    }  
  ]  
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[驗證 Auto Scaling 群組的擴展活動](#)。

範例 2：描述已刪除群組的擴展活動

若要描述刪除 Auto Scaling 群組之後的擴展活動，請新增 `--include-deleted-groups` 選項。

```
aws autoscaling describe-scaling-activities \  
  --auto-scaling-group-name my-asg \  
  --include-deleted-groups
```

輸出：

```
{  
  "Activities": [  

```

```

    {
      "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
      "Description": "Launching a new EC2 instance.  Status Reason: Your
Spot request price of 0.001 is lower than the minimum required Spot request
fulfillment price of 0.0031. Launching EC2 instance failed.",
      "AutoScalingGroupName": "my-asg",
      "Cause": "At 2021-01-13T20:47:24Z a user request update of
AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired
capacity from 0 to 3.  At 2021-01-13T20:47:27Z an instance was started in
response to a difference between desired and actual capacity, increasing the
capacity from 0 to 3.",
      "StartTime": "2021-01-13T20:47:30.094Z",
      "EndTime": "2021-01-13T20:47:30Z",
      "StatusCode": "Failed",
      "StatusMessage": "Your Spot request price of 0.001 is lower than
the minimum required Spot request fulfillment price of 0.0031. Launching EC2
instance failed.",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}",
      "AutoScalingGroupState": "Deleted",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    }
  ]
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的故障診斷](#)。

範例 3：描述指定數量的擴展活動

若要傳回特定數量的活動，請使用 `--max-items` 選項。

```

aws autoscaling describe-scaling-activities \
  --max-items 1

```

輸出：

```

{
  "Activities": [
    {

```

```

        "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
        "Description": "Launching a new EC2 instance: i-0d44425630326060f",
        "AutoScalingGroupName": "my-asg",
        "Cause": "At 2020-10-30T19:35:51Z a user request update of
AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
desired capacity from 0 to 16. At 2020-10-30T19:36:07Z an instance was started
in response to a difference between desired and actual capacity, increasing the
capacity from 0 to 16.",
        "StartTime": "2020-10-30T19:36:09.766Z",
        "EndTime": "2020-10-30T19:36:41Z",
        "StatusCode": "Successful",
        "Progress": 100,
        "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
    }
]
}

```

如果輸出包含 `NextToken` 欄位，則會有更多活動。若要取得其他活動，請在後續呼叫時使用此欄位的值和 `--starting-token` 選項，如下所示。

```

aws autoscaling describe-scaling-activities \
  --starting-token Z3M3LMPEXAMPLE

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[驗證 Auto Scaling 群組的擴展活動](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeScalingActivities](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {

```



```
try {
    DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
        .autoScalingGroupName(groupName)
        .maxRecords(10)
        .build();

    DescribeScalingActivitiesResponse response = autoScalingClient
        .describeScalingActivities(scalingActivitiesRequest);
    List<Activity> activities = response.activities();
    for (Activity activity : activities) {
        System.out.println("The activity Id is " +
activity.activityId());
        System.out.println("The activity details are " +
activity.details());
    }

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DescribeScalingActivities](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }
```

```
        maxRecords = 10
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
${group.healthCheckType}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DescribeScalingActivities](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [DescribeScalingActivities](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例描述指定的 Auto Scaling 群組過去六週的擴展活動。

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

輸出：

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
                      group desired capacity changing the desired
                      capacity from 1 to 2. At 2015-11-22T15:45:34Z an instance
                      was started in response to a difference
                      between desired and actual capacity, increasing the
                      capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress              : 100
StartTime             : 11/22/2015 7:45:35 AM
StatusCode            : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
                      AutoScalingGroup changing the desired capacity
                      from 0 to 1. At 2015-11-20T22:57:58Z an instance was
                      started in response to a difference betwe
                      en desired and actual capacity, increasing the capacity
                      from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress              : 100
StartTime             : 11/20/2015 2:57:59 PM
StatusCode            : Successful
StatusMessage        :
```

範例 2：此範例描述指定的擴展活動。

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

範例 3：此範例描述所有 Auto Scaling 群組過去六週的擴展活動。

```
Get-ASScalingActivity
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考(V4)》中的 [DescribeScalingActivities](#)。

Tools for PowerShell V5

範例 1：此範例描述指定的 Auto Scaling 群組過去六週的擴展活動。

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

輸出：

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
                      group desired capacity changing the desired
                      capacity from 1 to 2. At 2015-11-22T15:45:34Z an instance
                      was started in response to a difference
                      between desired and actual capacity, increasing the
                      capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress              : 100
StartTime             : 11/22/2015 7:45:35 AM
StatusCode            : Successful
StatusMessage         :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
                      AutoScalingGroup changing the desired capacity
                      from 0 to 1. At 2015-11-20T22:57:58Z an instance was
                      started in response to a difference betwe
```

```

        en desired and actual capacity, increasing the capacity
        from 0 to 1.
Description      : Launching a new EC2 instance: i-93633f9b
Details          : {"Availability Zone":"us-west-2b","Subnet
ID":"subnet-5264e837"}
EndTime         : 11/20/2015 2:58:32 PM
Progress        : 100
StartTime       : 11/20/2015 2:57:59 PM
StatusCode      : Successful
StatusMessage   :

```

範例 2：此範例描述指定的擴展活動。

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

範例 3：此範例描述所有 Auto Scaling 群組過去六週的擴展活動。

```
Get-ASScalingActivity
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeScalingActivities](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

```

```

def describe_scaling_activities(self, group_name: str) -> List[Dict[str,
Any]]:
    """
    Gets information about scaling activities for the group. Scaling
    activities
    are things like instances stopping or starting in response to user
    requests
    or capacity changes.

    :param group_name: The name of the group to look up.
    :return: A list of dictionaries representing the scaling activities for
    the
        group, ordered with the most recent activity first.
    :raises ClientError: If there is an error describing the scaling
    activities.
    """
    try:
        paginator = self.autoscaling_client.get_paginator(
            "describe_scaling_activities"
        )
        response_iterator =
paginator.paginate(AutoScalingGroupName=group_name)
        activities = []
        for response in response_iterator:
            activities.extend(response.get("Activities", []))

        logger.info(
            f"Successfully described scaling activities for group
            '{group_name}'."
        )

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(
            f"Couldn't describe scaling activities for group '{group_name}'.
            Error code: {error_code}, Message: {err.response['Error']['Message']}"
        )

        if error_code == "ResourceContentionFault":
            logger.error(
                f"There is a conflict with another operation that is
                modifying the Auto Scaling group '{group_name}'."
            )

```

```

        "Please try again later."
    )
    raise
else:
    return activities

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [DescribeScalingActivities](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
    let group = self
        .autoscaling
        .describe_auto_scaling_groups()
        .auto_scaling_group_names(self.auto_scaling_group_name.clone())
        .send()
        .await
        .map(|s| {
            s.auto_scaling_groups()
                .iter()
                .map(|s| {
                    format!(
                        "{}: {}",
                        s.auto_scaling_group_name().unwrap_or("Unknown"),
                        s.status().unwrap_or("Unknown")
                    )
                })
                .collect::<Vec<String>>()
        })
        .map_err(|e| {

```

```

        ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
    });

    let instances = self
        .list_instances()
        .await
        .map_err(|e| anyhow!("There was an error listing instances: {e}",));

    // 10. DescribeScalingActivities: list the scaling activities that have
    occurred for the group so far.
    // Bonus: use CloudWatch API to get and show some metrics collected for
    the group.
    // CW.ListMetrics with Namespace='AWS/AutoScaling' and
    Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
    // CW.GetMetricStatistics with Statistics='Sum'. Start and End times
    must be in UTC!
    let activities = self
        .autoscaling
        .describe_scaling_activities()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .into_paginator()
        .items()
        .send()
        .collect::()
        .await
        .map_err(|e| {
            anyhow!(
                "There was an error retrieving scaling activities: {}",
                DisplayErrorContext(&e)
            )
        });

    AutoScalingScenarioDescription {
        group,
        instances,
        activities,
    }
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [DescribeScalingActivities](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
  DATA(lo_output) = ao_asc->describescalingactivities(
    iv_autoscalinggroupname = iv_group_name ).

  ot_output = lo_output->get_activities( ).

  MESSAGE 'Scaling activities retrieved successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DescribeScalingActivities](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeScalingProcessTypes** 與 CLI

下列程式碼範例示範如何使用 DescribeScalingProcessTypes。

CLI

AWS CLI

描述可用的程序類型

此範例描述可用的程序類型。

```
aws autoscaling describe-scaling-process-types
```

輸出：

```
{
  "Processes": [
    {
      "ProcessName": "AZRebalance"
    },
    {
      "ProcessName": "AddToLoadBalancer"
    },
    {
      "ProcessName": "AlarmNotification"
    },
    {
      "ProcessName": "HealthCheck"
    },
    {
      "ProcessName": "InstanceRefresh"
    },
    {
      "ProcessName": "Launch"
    },
    {
      "ProcessName": "ReplaceUnhealthy"
    },
    {
      "ProcessName": "ScheduledActions"
    },
    {
      "ProcessName": "Terminate"
    }
  ]
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[暫停及繼續擴展程序](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeScalingProcessTypes](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例列出 Auto Scaling 支援的程序類型。

```
Get-ASScalingProcessType
```

輸出：

```
ProcessName
-----
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeScalingProcessTypes](#)。

Tools for PowerShell V5

範例 1：此範例列出 Auto Scaling 支援的程序類型。

```
Get-ASScalingProcessType
```

輸出：

```
ProcessName
-----
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
```

```
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeScalingProcessTypes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeScheduledActions** 與 CLI

下列程式碼範例示範如何使用 `DescribeScheduledActions`。

CLI

AWS CLI

範例 1：描述所有排程動作

此範例描述所有排程動作。

```
aws autoscaling describe-scheduled-actions
```

輸出：

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
```

```

        "DesiredCapacity": 4,
        "TimeZone": "America/New_York"
    }
]
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

範例 2：描述指定群組的排程動作

若要描述特定 Auto Scaling 群組的排程動作，請使用 `--auto-scaling-group-name` 選項。

```

aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg

```

輸出：

```

{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

範例 3：描述指定的排程動作

若要描述特定的排程動作，請使用 `--scheduled-action-names` 選項。

```
aws autoscaling describe-scheduled-actions \  
--scheduled-action-names my-recurring-action
```

輸出：

```
{  
  "ScheduledUpdateGroupActions": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "ScheduledActionName": "my-recurring-action",  
      "Recurrence": "30 0 1 1,6,12 *",  
      "ScheduledActionARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-  
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-  
action",  
      "StartTime": "2023-12-01T04:00:00Z",  
      "Time": "2023-12-01T04:00:00Z",  
      "MinSize": 1,  
      "MaxSize": 6,  
      "DesiredCapacity": 4,  
      "TimeZone": "America/New_York"  
    }  
  ]  
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

範例 4：描述具有指定開始時間的排程動作

若要描述在特定時間開始的排程動作，請使用 `--start-time` 選項。

```
aws autoscaling describe-scheduled-actions \  
--start-time "2023-12-01T04:00:00Z"
```

輸出：

```
{  
  "ScheduledUpdateGroupActions": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "ScheduledActionName": "my-recurring-action",
```

```

        "Recurrence": "30 0 1 1,6,12 *",
        "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
        "StartTime": "2023-12-01T04:00:00Z",
        "Time": "2023-12-01T04:00:00Z",
        "MinSize": 1,
        "MaxSize": 6,
        "DesiredCapacity": 4,
        "TimeZone": "America/New_York"
    }
]
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

範例 5：描述在指定時間結束的排程動作

若要描述在特定時間結束的排程動作，請使用 `--end-time` 選項。

```

aws autoscaling describe-scheduled-actions \
    --end-time "2023-12-01T04:00:00Z"

```

輸出：

```

{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}

```

```
    }
  ]
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

範例 6：描述指定數量的排程動作

若要傳回特定數量的排程動作，請使用 `--max-items` 選項。

```
aws autoscaling describe-scheduled-actions \
  --auto-scaling-group-name my-asg \
  --max-items 1
```

輸出：

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}
```

如果輸出包含 `NextToken` 欄位，則會有更多排程動作。若要取得其他排程動作，請在後續呼叫時使用此欄位的值和 `--starting-token` 選項，如下所示。

```
aws autoscaling describe-scheduled-actions \
  --auto-scaling-group-name my-asg \
```



```
--starting-token Z3M3LMPEXAMPLE
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeScheduledActions](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例描述指定的 Auto Scaling 群組的排程擴展動作。

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

輸出：

```
AutoScalingGroupName : my-asg
DesiredCapacity       : 10
EndTime               :
MaxSize               :
MinSize               :
Recurrence            :
ScheduledActionARN    : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
ScheduledActionName   : myScheduledAction
StartTime              : 11/30/2015 8:00:00 AM
Time                  : 11/30/2015 8:00:00 AM
```

範例 2：此範例描述指定的排程擴展動作。

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
"myScheduledScaleIn")
```

範例 3：此範例描述在指定時間開始的排程擴展動作。

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

範例 4：此範例描述在指定時間結束的排程擴展動作。

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

範例 5：此範例描述所有 Auto Scaling 群組的排程擴展動作。

```
Get-ASScheduledAction
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeScheduledActions](#)。

Tools for PowerShell V5

範例 1：此範例描述指定的 Auto Scaling 群組的排程擴展動作。

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

輸出：

```
AutoScalingGroupName : my-asg
DesiredCapacity       : 10
EndTime               :
MaxSize               :
MinSize               :
Recurrence             :
ScheduledActionARN    : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
ScheduledActionName   : myScheduledAction
StartTime              : 11/30/2015 8:00:00 AM
Time                  : 11/30/2015 8:00:00 AM
```

範例 2：此範例描述指定的排程擴展動作。

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
"myScheduledScaleIn")
```

範例 3：此範例描述在指定時間開始的排程擴展動作。

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

範例 4：此範例描述在指定時間結束的排程擴展動作。

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

範例 5：此範例描述所有 Auto Scaling 群組的排程擴展動作。

```
Get-ASScheduledAction
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeScheduledActions](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DescribeTags** 與 CLI

下列程式碼範例示範如何使用 DescribeTags。

CLI

AWS CLI

描述所有標籤

此範例描述您的所有標籤。

```
aws autoscaling describe-tags
```

輸出：

```
{
  "Tags": [
    {
      "ResourceType": "auto-scaling-group",
      "ResourceId": "my-asg",
      "PropagateAtLaunch": true,
      "Value": "Research",
      "Key": "Dept"
    },
  ],
}
```

```
{
  "ResourceType": "auto-scaling-group",
  "ResourceId": "my-asg",
  "PropagateAtLaunch": true,
  "Value": "WebServer",
  "Key": "Role"
}
]
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[標記 Auto Scaling 群組和執行個體](#)。

範例 2：描述指定群組的標籤

若要描述特定 Auto Scaling 群組的標籤，請使用 `--filters` 選項。

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[標記 Auto Scaling 群組和執行個體](#)。

範例 3：描述指定的標籤數量

若要傳回特定數量的標籤，請使用 `--max-items` 選項。

```
aws autoscaling describe-tags \
  --max-items 1
```

如果輸出包含 `NextToken` 欄位，則會有更多標籤。若要取得其他標籤，請在後續呼叫時使用此欄位的值和 `--starting-token` 選項，如下所示。

```
aws autoscaling describe-tags \
  --filters Name=auto-scaling-group,Values=my-asg \
  --starting-token Z3M3LMPEXAMPLE
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[標記 Auto Scaling 群組和執行個體](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeTags](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例描述鍵值為 'myTag' 或 'myTag2' 的標籤。篩選條件名稱的可能值為 'auto-scaling-group'、'key'、'value' 和 'propagate-at-launch'。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

輸出：

```
Key           : myTag2
PropagateAtLaunch : True
ResourceId     : my-asg
ResourceType   : auto-scaling-group
Value         : myTagValue2

Key           : myTag
PropagateAtLaunch : True
ResourceId     : my-asg
ResourceType   : auto-scaling-group
Value         : myTagValue
```

範例 2：使用 PowerShell 版本 2 時，必須使用 New-Object 來建立每個篩選條件參數的篩選條件。

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

範例 3：此範例描述所有 Auto Scaling 群組的所有標籤。

```
Get-ASTag
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeTags](#)。

Tools for PowerShell V5

範例 1：此範例描述鍵值為 'myTag' 或 'myTag2' 的標籤。篩選條件名稱的可能值為 'auto-scaling-group'、'key'、'value' 和 'propagate-at-launch'。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

輸出：

```
Key           : myTag2
PropagateAtLaunch : True
ResourceId     : my-asg
ResourceType   : auto-scaling-group
Value         : myTagValue2

Key           : myTag
PropagateAtLaunch : True
ResourceId     : my-asg
ResourceType   : auto-scaling-group
Value         : myTagValue
```

範例 2：使用 PowerShell 版本 2 時，必須使用 New-Object 來建立每個篩選條件參數的篩選條件。

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

範例 3：此範例描述所有 Auto Scaling 群組的所有標籤。

```
Get-ASTag
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeTags](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 `DescribeTerminationPolicyTypes` 與 CLI

下列程式碼範例示範如何使用 `DescribeTerminationPolicyTypes`。

CLI

AWS CLI

描述可用的終止政策類型

此範例描述可用的終止政策類型。

```
aws autoscaling describe-termination-policy-types
```

輸出：

```
{
  "TerminationPolicyTypes": [
    "AllocationStrategy",
    "ClosestToNextInstanceHour",
    "Default",
    "NewestInstance",
    "OldestInstance",
    "OldestLaunchConfiguration",
    "OldestLaunchTemplate"
  ]
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [控制在縮減期間終止的 Auto Scaling 執行個體](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeTerminationPolicyTypes](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例列出 Auto Scaling 支援的終止政策。

```
Get-ASTerminationPolicyType
```

輸出：

```
ClosestToNextInstanceHour  
Default  
NewestInstance  
OldestInstance  
OldestLaunchConfiguration
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DescribeTerminationPolicyTypes](#)。

Tools for PowerShell V5

範例 1：此範例列出 Auto Scaling 支援的終止政策。

```
Get-ASTerminationPolicyType
```

輸出：

```
ClosestToNextInstanceHour  
Default  
NewestInstance  
OldestInstance  
OldestLaunchConfiguration
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DescribeTerminationPolicyTypes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DetachInstances** 與 CLI

下列程式碼範例示範如何使用 DetachInstances。

CLI

AWS CLI

將執行個體與 Auto Scaling 群組分開

此範例將指定的執行個體與指定的 Auto Scaling 群組分開。

```
aws autoscaling detach-instances \  
  --instance-ids i-030017cfa84b20135 \  
  --auto-scaling-group-name my-asg \  
  --should-decrement-desired-capacity
```

輸出：

```
{  
  "Activities": [  
    {  
      "ActivityId": "5091cb52-547a-47ce-a236-c9ccbc2cb2c9",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Detaching EC2 instance: i-030017cfa84b20135",  
      "Cause": "At 2020-10-31T17:35:04Z instance i-030017cfa84b20135 was  
detached in response to a user request, shrinking the capacity from 2 to 1.",  
      "StartTime": "2020-04-12T15:02:16.179Z",  
      "StatusCode": "InProgress",  
      "Progress": 50,  
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":  
\"us-west-2c\"}"  
    }  
  ]  
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DetachInstances](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的執行個體與指定的 Auto Scaling 群組分開，並減少所需的容量，如此 Auto Scaling 就不會啟動替代執行個體。

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

輸出：

```
ActivityId          : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause               : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
                     in response to a user request, shrinking
                       the capacity from 2 to 1.
Description         : Detaching EC2 instance: i-93633f9b
Details             : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime             :
Progress            : 50
StartTime           : 11/20/2015 2:34:59 PM
StatusCode          : InProgress
StatusMessage       :
```

範例 2：此範例將指定的執行個體與指定的 Auto Scaling 群組分開，而不會減少所需的容量。Auto Scaling 會啟動替代執行個體。

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

輸出：

```
ActivityId          : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause               : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
                     in response to a user request.
Description         : Detaching EC2 instance: i-7bf746a2
Details             : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime             :
Progress            : 50
StartTime           : 11/20/2015 2:34:59 PM
StatusCode          : InProgress
StatusMessage       :
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DetachInstances](#)。

Tools for PowerShell V5

範例 1：此範例將指定的執行個體與指定的 Auto Scaling 群組分開，並減少所需的容量，如此 Auto Scaling 就不會啟動替代執行個體。

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

輸出：

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
                      in response to a user request, shrinking
                      the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress              : 50
StartTime             : 11/20/2015 2:34:59 PM
StatusCode            : InProgress
StatusMessage         :
```

範例 2：此範例將指定的執行個體與指定的 Auto Scaling 群組分開，而不會減少所需的容量。Auto Scaling 會啟動替代執行個體。

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

輸出：

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
                      in response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
```

```
EndTime           :  
Progress          : 50  
StartTime         : 11/20/2015 2:34:59 PM  
StatusCode        : InProgress  
StatusMessage     :
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DetachInstances](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **DetachLoadBalancers** 與 CLI

下列程式碼範例示範如何使用 DetachLoadBalancers。

CLI

AWS CLI

將 Classic Load Balancer 與 Auto Scaling 群組分開

此範例將指定的 Classic Load Balancer 與指定的 Auto Scaling 群組分開。

```
aws autoscaling detach-load-balancers \  
  --load-balancer-names my-load-balancer \  
  --auto-scaling-group-name my-asg
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [將負載平衡器附加到您的 Auto Scaling 群組](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DetachLoadBalancers](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的負載平衡器與指定的 Auto Scaling 群組分開。

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DetachLoadBalancers](#)。

Tools for PowerShell V5

範例 1：此範例將指定的負載平衡器與指定的 Auto Scaling 群組分開。

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DetachLoadBalancers](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DisableMetricsCollection 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DisableMetricsCollection。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>
/// Disable the collection of metric data for an Amazon EC2 Auto Scaling
/// group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
```

```

    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
        _amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [DisableMetricsCollection](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
request.SetAutoScalingGroupName(groupName);

Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
    autoScalingClient.DisableMetricsCollection(request);

if (outcome.IsSuccess()) {

```

```
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DisableMetricsCollection](#)。

CLI

AWS CLI

停用 Auto Scaling 群組的指標收集

此範例會停用指定的 Auto Scaling 群組的 GroupDesiredCapacity 指標收集。

```
aws autoscaling disable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [監控 Auto Scaling 群組和執行個體的 CloudWatch 指標](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DisableMetricsCollection](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DisableMetricsCollection](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }
}
```



```
AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
    autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
    println("The disable metrics collection operation was successful")
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DisableMetricsCollection](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function disableMetricsCollection($autoScalingGroupName)
{
    return $this->autoScalingClient->disableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [DisableMetricsCollection](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例停用指定的 Auto Scaling 群組的指定指標監控功能。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
@("GroupMinSize", "GroupMaxSize")
```

範例 2：此範例停用指定的 Auto Scaling 群組的所有指標監控功能。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [DisableMetricsCollection](#)。

Tools for PowerShell V5

範例 1：此範例停用指定的 Auto Scaling 群組的指定指標監控功能。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric  
@("GroupMinSize", "GroupMaxSize")
```

範例 2：此範例停用指定的 Auto Scaling 群組的所有指標監控功能。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [DisableMetricsCollection](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client
```

```

def disable_metrics(self, group_name: str) -> Dict[str, Any]:
    """
    Stops CloudWatch metric collection for the Auto Scaling group.

    :param group_name: The name of the group.
    :return: A dictionary with the response from disabling the metrics
collection.
    :raises ClientError: If there is an error disabling metrics collection.
    """
    try:
        response = self.autoscaling_client.disable_metrics_collection(
            AutoScalingGroupName=group_name
        )
        logger.info(
            f"Successfully disabled metrics collection for group
'{group_name}'."
        )
        return response
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(
            f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
        )

        if error_code == "ResourceContentionFault":
            logger.error(
                f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                "Please try again later."
            )
            raise

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [DisableMetricsCollection](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
// If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
let _ = self
    .autoscaling
    .disable_metrics_collection()
    .auto_scaling_group_name(self.auto_scaling_group_name.clone())
    .send()
    .await;
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [DisableMetricsCollection](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
    ao_asc->disablemetricscollection(
        iv_autoscalinggroupname = iv_group_name ).

    MESSAGE 'Metrics collection disabled successfully' TYPE 'I'.
```

```

CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DisableMetricsCollection](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

EnableMetricsCollection 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 EnableMetricsCollection。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/// <summary>
/// Enable the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> EnableMetricsCollectionAsync(string groupName)
{
    var listMetrics = new List<string>

```

```
        {
            "GroupMaxSize",
        };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
            _amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [EnableMetricsCollection](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
request.SetAutoScalingGroupName(groupName);

request.AddMetrics("GroupMinSize");
```

```
request.AddMetrics("GroupMaxSize");
request.AddMetrics("GroupDesiredCapacity");
request.AddMetrics("GroupInServiceInstances");
request.AddMetrics("GroupTotalInstances");
request.SetGranularity("1Minute");

Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
    autoScalingClient.EnableMetricsCollection(request);
if (outcome.IsSuccess()) {
    std::cout << "Auto Scaling metrics have been enabled."
              << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [EnableMetricsCollection](#)。

CLI

AWS CLI

範例 1：啟用 Auto Scaling 群組的指標收集

此範例啟用指定的 Auto Scaling 群組的資料收集。

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --granularity "1Minute"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[監控 Auto Scaling 群組和執行個體的 CloudWatch 指標](#)。

範例 2：收集 Auto Scaling 群組的指定指標的資料

若要收集特定指標的資料，請使用 `--metrics` 選項。

```
aws autoscaling enable-metrics-collection \
  --auto-scaling-group-name my-asg \
  --metrics GroupDesiredCapacity --granularity "1Minute"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[監控 Auto Scaling 群組和執行個體的 CloudWatch 指標](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [EnableMetricsCollection](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
        EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```


- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [EnableMetricsCollection](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- 如需 API 詳細資訊，請參閱[適用於 Kotlin 的 AWS SDK API 參考](#)中的 [EnableMetricsCollection](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function enableMetricsCollection($autoScalingGroupName, $granularity)
{
    return $this->autoScalingClient->enableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'Granularity' => $granularity,
    ]);
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [EnableMetricsCollection](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例啟用指定的 Auto Scaling 群組的指定指標監控功能。

```
Enable-ASMetricsCollection -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

範例 2：此範例監控指定的 Auto Scaling 群組的所有指標監控功能。

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [EnableMetricsCollection](#)。

Tools for PowerShell V5

範例 1：此範例啟用指定的 Auto Scaling 群組的指定指標監控功能。

```
Enable-ASMetricsCollection -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

範例 2：此範例監控指定的 Auto Scaling 群組的所有指標監控功能。

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [EnableMetricsCollection](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str,
Any]:
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling
        activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        :return: A dictionary with the response from enabling the metrics
        collection.
        :raises ClientError: If there is an error enabling metrics collection.
        """
        try:
            response = self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
                Granularity="1Minute"
            )
            logger.info(
                f"Successfully enabled metrics for Auto Scaling group
                '{group_name}'."
            )
```

```
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    logger.error(
        f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
    )

    if error_code == "ResourceContentionFault":
        logger.error(
            f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
            "Please try again later."
        )
    elif error_code == "InvalidParameterCombination":
        logger.error(
            f"The combination of parameters provided for enabling metrics
on '{group_name}' is not valid. "
            "Please check the parameters and try again."
        )
    raise
else:
    return response
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [EnableMetricsCollection](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
let enable_metrics_collection = autoscaling
    .enable_metrics_collection()
    .auto_scaling_group_name(auto_scaling_group_name.as_str())
```

```

        .granularity("1Minute")
        .set_metrics(Some(vec![
            String::from("GroupMinSize"),
            String::from("GroupMaxSize"),
            String::from("GroupDesiredCapacity"),
            String::from("GroupInServiceInstances"),
            String::from("GroupTotalInstances"),
        ]))
        .send()
        .await;

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [EnableMetricsCollection](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

" Example: iv_group_name = 'my-auto-scaling-group'
" Example: it_metrics contains list of metrics like 'GroupMinSize',
'GroupMaxSize', etc.

TRY.
    ao_asc->enablemetricscollection(
        iv_autoscalinggroupname = iv_group_name
        it_metrics = it_metrics
        iv_granularity = '1Minute' ).

    MESSAGE 'Metrics collection enabled successfully' TYPE 'I'.

CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [EnableMetricsCollection](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **EnterStandby** 與 CLI

下列程式碼範例示範如何使用 EnterStandby。

CLI

AWS CLI

將執行個體移至待命模式

此範例將指定的執行個體置於待命模式。這對於更新或故障診斷目前服務中的執行個體非常有用。

```
aws autoscaling enter-standby \
  --instance-ids i-061c63c5eb45f0416 \
  --auto-scaling-group-name my-asg \
  --should-decrement-desired-capacity
```

輸出：

```
{
  "Activities": [
    {
      "ActivityId": "ffa056b4-6ed3-41ba-ae7c-249dfae6eba1",
      "AutoScalingGroupName": "my-asg",
      "Description": "Moving EC2 instance to Standby: i-061c63c5eb45f0416",
      "Cause": "At 2020-10-31T20:31:00Z instance i-061c63c5eb45f0416 was
moved to standby in response to a user request, shrinking the capacity from 1 to
0.",
      "StartTime": "2020-10-31T20:31:00.949Z",
      "StatusCode": "InProgress",
      "Progress": 50,
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
    }
  ]
}
```

```
    }
  ]
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 執行個體生命週期](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [EnterStandby](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的執行個體置於待命模式，並減少所需的容量，如此 Auto Scaling 就不會啟動替代執行個體。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

輸出：

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
                      standby in response to a user request,
                      shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress              : 50
StartTime             : 11/22/2015 7:48:06 AM
StatusCode            : InProgress
StatusMessage        :
```

範例 2：此範例將指定的執行個體置於待命模式，而不會減少所需的容量。Auto Scaling 會啟動替代執行個體。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

輸出：

```
ActivityId          : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause               : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
                    standby in response to a user request.
Description         : Moving EC2 instance to Standby: i-95b8484f
Details             : {"Availability Zone":"us-west-2b","Subnet
                    ID":"subnet-5264e837"}
EndTime            :
Progress            : 50
StartTime           : 11/22/2015 7:48:06 AM
StatusCode          : InProgress
StatusMessage       :
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [EnterStandby](#)。

Tools for PowerShell V5

範例 1：此範例將指定的執行個體置於待命模式，並減少所需的容量，如此 Auto Scaling 就不會啟動替代執行個體。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

輸出：

```
ActivityId          : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause               : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
                    standby in response to a user request,
                    shrinking the capacity from 2 to 1.
Description         : Moving EC2 instance to Standby: i-95b8484f
Details             : {"Availability Zone":"us-west-2b","Subnet
                    ID":"subnet-5264e837"}
EndTime            :
Progress            : 50
StartTime           : 11/22/2015 7:48:06 AM
StatusCode          : InProgress
StatusMessage       :
```


範例 2：此範例將指定的執行個體置於待命模式，而不會減少所需的容量。Auto Scaling 會啟動替代執行個體。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $false
```

輸出：

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649  
AutoScalingGroupName : my-asg  
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to  
standby in response to a user request.  
Description          : Moving EC2 instance to Standby: i-95b8484f  
Details              : {"Availability Zone":"us-west-2b","Subnet  
ID":"subnet-5264e837"}  
EndTime              :  
Progress              : 50  
StartTime             : 11/22/2015 7:48:06 AM  
StatusCode            : InProgress  
StatusMessage         :
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [EnterStandby](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **ExecutePolicy** 與 CLI

下列程式碼範例示範如何使用 ExecutePolicy。

CLI

AWS CLI

執行擴展政策

此範例針對指定的 Auto Scaling 群組執行名為 my-step-scale-out-policy 的擴展政策。

```
aws autoscaling execute-policy \
```

```
--auto-scaling-group-name my-asg \  
--policy-name my-step-scale-out-policy \  
--metric-value 95 \  
--breach-threshold 80
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[步進和簡易擴展政策](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[ExecutePolicy](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例為指定的 Auto Scaling 群組執行指定的政策。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

範例 2：此範例在等待冷卻時間告一段落之後，為指定的 Auto Scaling 群組執行指定的政策。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -  
HonorCooldown $true
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的[ExecutePolicy](#)。

Tools for PowerShell V5

範例 1：此範例為指定的 Auto Scaling 群組執行指定的政策。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

範例 2：此範例在等待冷卻時間告一段落之後，為指定的 Auto Scaling 群組執行指定的政策。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -  
HonorCooldown $true
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的[ExecutePolicy](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **ExitStandby** 與 CLI

下列程式碼範例示範如何使用 ExitStandby。

CLI

AWS CLI

將執行個體移出待命模式

此範例將指定的執行個體移出待命模式。

```
aws autoscaling exit-standby \
  --instance-ids i-061c63c5eb45f0416 \
  --auto-scaling-group-name my-asg
```

輸出：

```
{
  "Activities": [
    {
      "ActivityId": "142928e1-a2dc-453a-9b24-b85ad6735928",
      "AutoScalingGroupName": "my-asg",
      "Description": "Moving EC2 instance out of Standby:
i-061c63c5eb45f0416",
      "Cause": "At 2020-10-31T20:32:50Z instance i-061c63c5eb45f0416 was
moved out of standby in response to a user request, increasing the capacity from
0 to 1.",
      "StartTime": "2020-10-31T20:32:50.222Z",
      "StatusCode": "PreInService",
      "Progress": 30,
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
    }
  ]
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [從 Auto Scaling 群組暫時移除執行個體](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ExitStandby](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的執行個體移出待命模式。

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

輸出：

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
                      of standby in response to a user
                      request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [ExitStandby](#)。

Tools for PowerShell V5

範例 1：此範例將指定的執行個體移出待命模式。

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

輸出：

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
                      of standby in response to a user
                      request, increasing the capacity from 1 to 2.
```

```

Description      : Moving EC2 instance out of Standby: i-95b8484f
Details          : {"Availability Zone":"us-west-2b","Subnet
  ID":"subnet-5264e837"}
EndTime          :
Progress         : 30
StartTime        : 11/22/2015 7:51:21 AM
StatusCode       : PreInService
StatusMessage    :

```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [ExitStandby](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 PutLifecycleHook 與 CLI

下列程式碼範例示範如何使用 PutLifecycleHook。

CLI

AWS CLI

範例 1：建立 lifecycle hook

此範例建立可在任何新啟動的執行個體上調用的 lifecycle hook，逾時時間為 4800 秒。這有助於將執行個體保持在等待狀態，直到使用者資料指令碼完成，或使用 EventBridge 叫用 AWS Lambda 函數。

```

aws autoscaling put-lifecycle-hook \
  --auto-scaling-group-name my-asg \
  --lifecycle-hook-name my-launch-hook \
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
  --heartbeat-timeout 4800

```

此命令不會產生輸出。如果具有相同名稱的 lifecycle hook 已存在，則會被新的 lifecycle hook 覆寫。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling lifecycle hook](#)。

範例 2：傳送 Amazon SNS 電子郵件訊息通知您執行個體狀態轉移

此範例建立具有 Amazon SNS 主題和 IAM 角色的 lifecycle hook，以用於在執行個體啟動時接收通知。

```
aws autoscaling put-lifecycle-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-hook-name my-launch-hook \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \  
  --notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \  
  --role-arn arn:aws:iam::123456789012:role/my-auto-scaling-role
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling lifecycle hook](#)。

範例 3：發佈訊息至 Amazon SQS 佇列

此範例會建立 lifecycle hook，將具有中繼資料的訊息發佈至指定的 Amazon SQS 佇列。

```
aws autoscaling put-lifecycle-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-hook-name my-launch-hook \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \  
  --notification-target-arn arn:aws:sqs:us-west-2:123456789012:my-sqs-queue \  
  --role-arn arn:aws:iam::123456789012:role/my-notification-role \  
  --notification-metadata "SQS message metadata"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling lifecycle hook](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [PutLifecycleHook](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的 lifecycle hook 新增至指定的 Auto Scaling 群組。

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
  "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
```

```
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN  
"arn:aws:iam::123456789012:role/my-iam-role"
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [PutLifecycleHook](#)。

Tools for PowerShell V5

範例 1：此範例將指定的 lifecycle hook 新增至指定的 Auto Scaling 群組。

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
"myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -  
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN  
"arn:aws:iam::123456789012:role/my-iam-role"
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [PutLifecycleHook](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 PutNotificationConfiguration 與 CLI

下列程式碼範例示範如何使用 PutNotificationConfiguration。

CLI

AWS CLI

新增通知

此範例將指定的通知新增至指定的 Auto Scaling 群組。

```
aws autoscaling put-notification-configuration \  
  --auto-scaling-group-name my-asg \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \  
  --notification-type autoscaling:TEST_NOTIFICATION
```

此命令不會產生輸出。

有關詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的 [取得 Auto Scaling 群組擴展時的 Amazon SNS 通知](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [PutNotificationConfiguration](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的 Auto Scaling 群組設定為在啟動 EC2 執行個體時，傳送通知至指定的 SNS 主題。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

範例 2：此範例將指定的 Auto Scaling 群組設定為在啟動或終止 EC2 執行個體時，傳送通知至指定的 SNS 主題。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
@("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [PutNotificationConfiguration](#)。

Tools for PowerShell V5

範例 1：此範例將指定的 Auto Scaling 群組設定為在啟動 EC2 執行個體時，傳送通知至指定的 SNS 主題。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

範例 2：此範例將指定的 Auto Scaling 群組設定為在啟動或終止 EC2 執行個體時，傳送通知至指定的 SNS 主題。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
@("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [PutNotificationConfiguration](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **PutScalingPolicy** 與 CLI

下列程式碼範例示範如何使用 PutScalingPolicy。

CLI

AWS CLI

將目標追蹤擴展政策新增至 Auto Scaling 群組

下列 put-scaling-policy 範例將目標追蹤擴展政策套用至指定的 Auto Scaling 群組。該輸出包含了系統代您建立的兩個 CloudWatch 警示的 ARN 及名稱。如果已存在具有相同名稱的擴展政策，其會被新擴展政策覆寫。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
  --policy-name alb1000-target-tracking-scaling-policy \
  --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

config.json 的內容：

```
{
  "TargetValue": 1000.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ALBRequestCountPerTarget",
    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
  }
}
```

輸出：

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-
c665-4bfd-aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/alb1000-
target-tracking-scaling-policy",
  "Alarms": [
    {
```

```

        "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
        "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
    },
    {
        "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
        "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }
]
}

```

如需更多範例，請參閱《Amazon EC2 Auto [Scaling 使用者指南](#)》中的 [AWS 命令列界面 \(AWS CLI\) 的範例擴展政策](#)。Amazon EC2 Auto Scaling

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [PutScalingPolicy](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定的政策新增至指定的 Auto Scaling 群組。指定的調整類型決定如何解譯 ScalingAdjustment 參數。使用 'ChangeInCapacity' 時，正值會增加指定的執行個體數量的容量，負值則會減少指定的執行個體數量的容量。

```

Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
"ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1

```

輸出：

```

arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy

```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [PutScalingPolicy](#)。

Tools for PowerShell V5

範例 1：此範例將指定的政策新增至指定的 Auto Scaling 群組。指定的調整類型決定如何解譯 ScalingAdjustment 參數。使用 'ChangeInCapacity' 時，正值會增加指定的執行個體數量的容量，負值則會減少指定的執行個體數量的容量。

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType  
"ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

輸出：

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-  
e1d769fc24ef:autoScalingGroupName/my-asg  
:policyName/myScaleInPolicy
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [PutScalingPolicy](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 PutScheduledUpdateGroupAction 與 CLI

下列程式碼範例示範如何使用 PutScheduledUpdateGroupAction。

CLI

AWS CLI

範例 1：將排程動作新增至 Auto Scaling 群組

此範例將指定的排程動作新增至指定的 Auto Scaling 群組。

```
aws autoscaling put-scheduled-update-group-action \  
  --auto-scaling-group-name my-asg \  
  --scheduled-action-name my-scheduled-action \  
  --start-time "2023-05-12T08:00:00Z" \  
  --min-size 2 \  
  --max-size 6 \  
  --desired-capacity 4
```

此命令不會產生輸出。如果具有相同名稱的排程動作已存在，則會被新的排程動作覆寫。

如需更多範例，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

範例 2：指定週期性排程

此範例會建立排程動作，以根據排定於每年 1 月、6 月和 12 月第一天的 00:30 執行的週期性排程進行擴展。

```
aws autoscaling put-scheduled-update-group-action \  
  --auto-scaling-group-name my-asg \  
  --scheduled-action-name my-recurring-action \  
  --recurrence "30 0 1 1,6,12 *" \  
  --min-size 2 \  
  --max-size 6 \  
  --desired-capacity 4
```

此命令不會產生輸出。如果具有相同名稱的排程動作已存在，則會被新的排程動作覆寫。

如需更多範例，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [PutScheduledUpdateGroupAction](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例會建立或更新一次性排程動作，以在指定的開始時間變更所需的容量。

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -  
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -  
DesiredCapacity 10
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [PutScheduledUpdateGroupAction](#)。

Tools for PowerShell V5

範例 1：此範例會建立或更新一次性排程動作，以在指定的開始時間變更所需的容量。

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -  
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -  
DesiredCapacity 10
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [PutScheduledUpdateGroupAction](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **RecordLifecycleActionHeartbeat** 與 CLI

下列程式碼範例示範如何使用 `RecordLifecycleActionHeartbeat`。

CLI

AWS CLI

記錄生命週期動作活動訊號

此範例會記錄生命週期動作活動訊號，讓執行個體保持待定狀態。

```
aws autoscaling record-lifecycle-action-heartbeat \  
  --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling lifecycle hook](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [RecordLifecycleActionHeartbeat](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例記錄指定的生命週期動作的活動訊號。這會使執行個體保持待定狀態，直到您完成自訂動作為止。

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [RecordLifecycleActionHeartbeat](#)。

Tools for PowerShell V5

範例 1：此範例記錄指定的生命週期動作的活動訊號。這會使執行個體保持待定狀態，直到您完成自訂動作為止。

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [RecordLifecycleActionHeartbeat](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **ResumeProcesses** 與 CLI

下列程式碼範例示範如何使用 `ResumeProcesses`。

CLI

AWS CLI

恢復暫停的處理程序

此範例恢復指定的 Auto Scaling 群組的指定暫停擴展程序。

```
aws autoscaling resume-processes \  
  --auto-scaling-group-name my-asg \  
  --scaling-processes AlarmNotification
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [暫停及繼續擴展程序](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ResumeProcesses](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例恢復指定的 Auto Scaling 群組的指定 Auto Scaling 程序。

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

範例 2：此範例恢復指定的 Auto Scaling 群組的所有暫停 Auto Scaling 處理程序。

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [ResumeProcesses](#)。

Tools for PowerShell V5

範例 1：此範例恢復指定的 Auto Scaling 群組的指定 Auto Scaling 程序。

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

範例 2：此範例恢復指定的 Auto Scaling 群組的所有暫停 Auto Scaling 處理程序。

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [ResumeProcesses](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

SetDesiredCapacity 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 SetDesiredCapacity。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>
/// Set the desired capacity of an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="desiredCapacity">The desired capacity for the Auto
/// Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> SetDesiredCapacityAsync(
    string groupName,
    int desiredCapacity)
{
    var capacityRequest = new SetDesiredCapacityRequest
    {
        AutoScalingGroupName = groupName,
        DesiredCapacity = desiredCapacity,
    };

    var response = await
        _amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
    Console.WriteLine($"You have set the DesiredCapacity to
        {desiredCapacity}.");

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [SetDesiredCapacity](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
request.SetAutoScalingGroupName(groupName);
request.SetDesiredCapacity(2);

Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
    autoScalingClient.SetDesiredCapacity(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                << outcome.GetError().GetMessage()
                << std::endl;
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [SetDesiredCapacity](#)。

CLI

AWS CLI

為 Auto Scaling 群組設定所需的容量

此範例為指定的 Auto Scaling 群組設定所需的容量。

```
aws autoscaling set-desired-capacity \  
  --auto-scaling-group-name my-asg \  
  --desired-capacity 2 \  
  --honor-cooldown
```

如果成功，此命令會回到提示字元。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetDesiredCapacity](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,  
String groupName) {  
    try {  
        SetDesiredCapacityRequest capacityRequest =  
SetDesiredCapacityRequest.builder()  
            .autoScalingGroupName(groupName)  
            .desiredCapacity(2)  
            .build();  
  
        autoScalingClient.setDesiredCapacity(capacityRequest);  
        System.out.println("You have set the DesiredCapacity to 2");  
  
    } catch (AutoScalingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [SetDesiredCapacity](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun setDesiredCapacity(groupName: String) {  
    val capacityRequest =  
        SetDesiredCapacityRequest {  
            autoScalingGroupName = groupName  
            desiredCapacity = 2  
        }  
  
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
        autoScalingClient.setDesiredCapacity(capacityRequest)  
        println("You set the DesiredCapacity to 2")  
    }  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [SetDesiredCapacity](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)  
{  
    return $this->autoScalingClient->setDesiredCapacity([
```

```
        'AutoScalingGroupName' => $autoScalingGroupName,  
        'DesiredCapacity' => $desiredCapacity,  
    ]);  
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [SetDesiredCapacity](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例設定指定的 Auto Scaling 群組的大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

範例 2：此範例設定指定 Auto Scaling 群組的大小，並等待冷卻時間告一段落，然後再擴展到新的大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -  
HonorCooldown $true
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [SetDesiredCapacity](#)。

Tools for PowerShell V5

範例 1：此範例設定指定的 Auto Scaling 群組的大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

範例 2：此範例設定指定 Auto Scaling 群組的大小，並等待冷卻時間告一段落，然後再擴展到新的大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -  
HonorCooldown $true
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [SetDesiredCapacity](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
        keep the
        number of running instances equal to the desired capacity.

        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
                f"Successfully set desired capacity of {capacity} for Auto
                Scaling group '{group_name}'."
            )
```

```

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
            )
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'."
                    "Please wait for the activity to complete before attempting
to set the desired capacity."
                )
                logger.error(f"Full error:\n\t{err}")
            raise

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [SetDesiredCapacity](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
    // 7. SetDesiredCapacity: set desired capacity to 2.
    // Wait for a second instance to launch.
    let update_group = self
        .autoscaling
        .set_desired_capacity()
        .auto_scaling_group_name(self.auto_scaling_group_name.clone())
        .desired_capacity(capacity)
        .send()
}

```

```

        .await;
    if let Err(err) = update_group {
        return Err(ScenarioError::new(
            format!("Failed to update group to desired capacity
({capacity}))").as_str(),
            &err,
        ));
    }
    Ok(())
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [SetDesiredCapacity](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_capacity = 2

TRY.
    ao_asc->setdesiredcapacity(
        iv_autoscalinggroupname = iv_group_name
        iv_desiredcapacity = iv_capacity
        iv_honorcooldown = abap_false ).

    MESSAGE 'Desired capacity set successfully' TYPE 'I'.

    CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
        RAISE EXCEPTION lo_activity_in_progress.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [SetDesiredCapacity](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **SetInstanceHealth** 與 CLI

下列程式碼範例示範如何使用 SetInstanceHealth。

CLI

AWS CLI

設定執行個體的運作狀態

此範例設定指定之執行個體的運作狀態為 Unhealthy。

```
aws autoscaling set-instance-health \  
  --instance-id i-061c63c5eb45f0416 \  
  --health-status Unhealthy
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetInstanceHealth](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例將指定之執行個體的狀態設定為「運作狀態不良」，使其停止服務。Auto Scaling 會終止並取代執行個體。

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

範例 2：此範例將指定之執行個體的狀態設定為「運作狀態良好」，使其保持服務中狀態。Auto Scaling 群組的運作狀態檢查寬限期未遵循。

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -  
ShouldRespectGracePeriod $false
```


- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [SetInstanceHealth](#)。

Tools for PowerShell V5

範例 1：此範例將指定之執行個體的狀態設定為「運作狀態不良」，使其停止服務。Auto Scaling 會終止並取代執行個體。

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

範例 2：此範例將指定之執行個體的狀態設定為「運作狀態良好」，使其保持服務中狀態。Auto Scaling 群組的運作狀態檢查寬限期未遵循。

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -  
ShouldRespectGracePeriod $false
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [SetInstanceHealth](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **SetInstanceProtection** 與 CLI

下列程式碼範例示範如何使用 SetInstanceProtection。

CLI

AWS CLI

範例 1：啟用執行個體的執行個體保護設定

此範例啟用指定之執行個體的執行個體保護功能。

```
aws autoscaling set-instance-protection \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg --protected-from-scale-in
```

此命令不會產生輸出。

範例 2：停用執行個體的執行個體保護設定

此範例停用指定之執行個體的執行個體保護功能。

```
aws autoscaling set-instance-protection \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg \  
  --no-protected-from-scale-in
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetInstanceProtection](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例啟用指定之執行個體的執行個體保護功能。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $true
```

範例 2：此範例停用指定之執行個體的執行個體保護功能。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $false
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [SetInstanceProtection](#)。

Tools for PowerShell V5

範例 1：此範例啟用指定之執行個體的執行個體保護功能。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $true
```

範例 2：此範例停用指定之執行個體的執行個體保護功能。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $false
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [SetInstanceProtection](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配使用 **SuspendProcesses** 與 CLI

下列程式碼範例示範如何使用 SuspendProcesses。

CLI

AWS CLI

暫停 Auto Scaling 處理程序

此範例暫停指定的 Auto Scaling 群組的指定擴展程序。

```
aws autoscaling suspend-processes \  
  --auto-scaling-group-name my-asg \  
  --scaling-processes AlarmNotification
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [暫停及繼續擴展程序](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SuspendProcesses](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例暫停指定的 Auto Scaling 群組的指定 Auto Scaling 處理程序。

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess  
"AlarmNotification"
```

範例 2：此範例暫停指定的 Auto Scaling 群組的所有 Auto Scaling 處理程序。

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [SuspendProcesses](#)。

Tools for PowerShell V5

範例 1：此範例暫停指定的 Auto Scaling 群組的指定 Auto Scaling 處理程序。

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess  
"AlarmNotification"
```

範例 2：此範例暫停指定的 Auto Scaling 群組的所有 Auto Scaling 處理程序。

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [SuspendProcesses](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

TerminateInstanceInAutoScalingGroup 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 TerminateInstanceInAutoScalingGroup。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)
- [建置及管理彈性服務](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>  
/// Terminate all instances in the Auto Scaling group in preparation for
```

```
/// deleting the group.
/// </summary>
/// <param name="instanceId">The instance Id of the instance to terminate.</
param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
    string instanceId)
{
    var request = new TerminateInstanceInAutoScalingGroupRequest
    {
        InstanceId = instanceId,
        ShouldDecrementDesiredCapacity = false,
    };

    var response = await
        _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You have terminated the instance: {instanceId}");
        return true;
    }

    Console.WriteLine($"Could not terminate {instanceId}");
    return false;
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
    request.SetInstanceId(instanceIDs[instanceNumber - 1]);
    request.SetShouldDecrementDesiredCapacity(false);

    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
        autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Waiting for EC2 instance with ID '"
                    << instanceIDs[instanceNumber - 1] << "' to terminate..."
                    << std::endl;
    }
    else {
        std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }
}

```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

CLI

AWS CLI

終止 Auto Scaling 群組中的執行個體

此範例從指定的 Auto Scaling 群組終止指定的執行個體，而不更新群組的大小。Amazon EC2 Auto Scaling 會在指定的執行個體終止後啟動替代執行個體。

```
aws autoscaling terminate-instance-in-auto-scaling-group \
```

```
--instance-id i-061c63c5eb45f0416 \
--no-should-decrement-desired-capacity
```

輸出：

```
{
  "Activities": [
    {
      "ActivityId": "8c35d601-793c-400c-fcd0-f64a27530df7",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-061c63c5eb45f0416",
      "Cause": "",
      "StartTime": "2020-10-31T20:34:25.680Z",
      "StatusCode": "InProgress",
      "Progress": 0,
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":\
        \"us-west-2c\"}"
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
        TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
```

```
        .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

PHP

適用於 PHP 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public function terminateInstanceInAutoScalingGroup(
    $instanceId,
    $shouldDecrementDesiredCapacity = true,
    $attempts = 0
) {
    try {
        return $this->autoScalingClient-
>terminateInstanceInAutoScalingGroup([
            'InstanceId' => $instanceId,
            'ShouldDecrementDesiredCapacity' =>
$shouldDecrementDesiredCapacity,
        ]);
    } catch (AutoScalingException $exception) {
        if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
$attempts < 5) {
            error_log("Cannot terminate an instance while it is still
pending. Waiting then trying again.");
            sleep(5 * (1 + $attempts));
            return $this->terminateInstanceInAutoScalingGroup(
                $instanceId,
                $shouldDecrementDesiredCapacity,
                ++$attempts
            );
        } else {
            throw $exception;
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例終止了指定的執行個體，並減少其 Auto Scaling 群組所需的容量，如此 Auto Scaling 就不會啟動替代執行個體。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $true
```

輸出：

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5  
AutoScalingGroupName :  
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out  
                      of service in response to a user  
                      request, shrinking the capacity from 2 to 1.  
Description          : Terminating EC2 instance: i-93633f9b  
Details              : {"Availability Zone":"us-west-2b","Subnet  
                      ID":"subnet-5264e837"}  
EndTime              :  
Progress             : 0  
StartTime            : 11/22/2015 8:09:03 AM  
StatusCode           : InProgress  
StatusMessage        :
```

範例 2：此範例終止了指定的執行個體，但未減少其 Auto Scaling 群組所需的容量。Auto Scaling 會啟動替代執行個體。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $false
```

輸出：

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5  
AutoScalingGroupName :  
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out  
                      of service in response to a user  
                      request.  
Description          : Terminating EC2 instance: i-93633f9b  
Details              : {"Availability Zone":"us-west-2b","Subnet  
                      ID":"subnet-5264e837"}  
EndTime              :  
Progress             : 0  
StartTime            : 11/22/2015 8:09:03 AM  
StatusCode           : InProgress  
StatusMessage        :
```

```

EndTime           :
Progress          : 0
StartTime         : 11/22/2015 8:09:03 AM
StatusCode        : InProgress
StatusMessage     :

```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [TerminateInstanceInAutoScalingGroup](#)。

Tools for PowerShell V5

範例 1：此範例終止了指定的執行個體，並減少其 Auto Scaling 群組所需的容量，如此 Auto Scaling 就不會啟動替代執行個體。

```

Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true

```

輸出：

```

ActivityId        : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause             : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
                   of service in response to a user
                   request, shrinking the capacity from 2 to 1.
Description       : Terminating EC2 instance: i-93633f9b
Details           : {"Availability Zone":"us-west-2b","Subnet
                   ID":"subnet-5264e837"}
EndTime          :
Progress          : 0
StartTime         : 11/22/2015 8:09:03 AM
StatusCode        : InProgress
StatusMessage     :

```

範例 2：此範例終止了指定的執行個體，但未減少其 Auto Scaling 群組所需的容量。Auto Scaling 會啟動替代執行個體。

```

Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false

```

輸出：

```

ActivityId        : 2e40d9bd-1902-444c-abf3-6ea0002efdc5

```

```

AutoScalingGroupName :
Cause                  : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
                        of service in response to a user
                        request.
Description            : Terminating EC2 instance: i-93633f9b
Details                : {"Availability Zone":"us-west-2b","Subnet
                        ID":"subnet-5264e837"}
EndTime               :
Progress              : 0
StartTime              : 11/22/2015 8:09:03 AM
StatusCode             : InProgress
StatusMessage         :

```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [TerminateInstanceInAutoScalingGroup](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
    ) -> Dict[str, Any]:
        """
        Stops an instance.

```

```

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired
capacity
                                of the group. When passing True for this
parameter,
                                you can stop an instance without having a
replacement
                                instance start when the desired capacity
threshold is
                                crossed.
        :return: A dictionary containing details of the scaling activity that
occurs
                in response to this action.
        :raises ClientError: If there is an error terminating the instance.
        """
        try:
            response =
self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
ShouldDecrementDesiredCapacity=decrease_capacity
            )
            logger.info(f"Successfully terminated instance {instance_id}.")
            return response["Activity"]

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to terminate instance {instance_id}.")
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    "A scaling activity is currently in progress for the Auto
Scaling group "
                    f"associated with instance '{instance_id}'. "
                    "Please wait for the activity to complete before attempting
to terminate the instance."
                )
            elif error_code == "ResourceInUse":
                logger.error(
                    f"The instance '{instance_id}' or an associated resource is
currently in use "
                    "and cannot be terminated. "
                    "Ensure the instance is not involved in any ongoing processes
and try again."
                )

```

```
logger.error(f"Full error:\n\t{err}")
raise
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
    // Retrieve a list of instances in the auto scaling group.
    let auto_scaling_group = self.get_group().await?;
    let instances = auto_scaling_group.instances();
    // Or use other logic to find an instance to terminate.
    let instance = instances.first();
    if let Some(instance) = instance {
        let instance_id = if let Some(instance_id) = instance.instance_id() {
            instance_id
        } else {
            return Err(ScenarioError::with("Missing instance id"));
        };
        let termination = self
            .ec2
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await;
        if let Err(err) = termination {
            Err(ScenarioError::new(
                "There was a problem terminating an instance",
                &err,
            ))
        }
    }
}
```

```

        } else {
            Ok(())
        }
    } else {
        Err(ScenarioError::with("There was no instance to terminate"))
    }
}

async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
    let describe_auto_scaling_groups = self
        .autoscaling
        .describe_auto_scaling_groups()
        .auto_scaling_group_names(self.auto_scaling_group_name.clone())
        .send()
        .await;

    if let Err(err) = describe_auto_scaling_groups {
        return Err(ScenarioError::new(
            format!(
                "Failed to get status of autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )
            .as_str(),
            &err,
        ));
    }

    let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
    let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
    let auto_scaling_group = auto_scaling_groups.first();

    if auto_scaling_group.is_none() {
        return Err(ScenarioError::with(format!(
            "Could not find autoscaling group {}",
            self.auto_scaling_group_name.clone()
        )));
    }

    Ok(auto_scaling_group.unwrap().clone())
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
" Example: iv_instance_id = 'i-1234567890abcdef0'
" Example: iv_decrease_capacity = abap_true

TRY.
    DATA(lo_output) = ao_asc->terminateinstinautoscgroup(
        iv_instanceid = iv_instance_id
        iv_shoulddecrementdesiredcap = iv_decrease_capacity ).

    oo_output = lo_output->get_activity( ).

    MESSAGE 'Instance terminated successfully' TYPE 'I'.

CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [TerminateInstanceInAutoScalingGroup](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

UpdateAutoScalingGroup 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 UpdateAutoScalingGroup。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)
- [建置及管理彈性服務](#)

.NET

適用於 .NET 的 SDK (v4)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <summary>
/// Update the capacity of an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
/// <param name="maxSize">The maximum number of instances that can be
/// created for the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> UpdateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    int maxSize)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var groupRequest = new UpdateAutoScalingGroupRequest
    {
```

```

        MaxSize = maxSize,
        AutoScalingGroupName = groupName,
        LaunchTemplate = templateSpecification,
    };

    var response = await
        _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
        return true;
    }
    else
    {
        return false;
    }
}

```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [UpdateAutoScalingGroup](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;

```

```
request.SetAutoScalingGroupName(groupName);
request.SetMaxSize(3);

Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
    autoScalingClient.UpdateAutoScalingGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [UpdateAutoScalingGroup](#)。

CLI

AWS CLI

範例 1：更新 Auto Scaling 群組的大小限制

此範例將指定的 Auto Scaling 群組更新為大小下限 2，上限為 10。

```
aws autoscaling update-auto-scaling-group \
  --auto-scaling-group-name my-asg \
  --min-size 2 \
  --max-size 10
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [設定 Auto Scaling 群組的預設限制](#)。

範例 2：新增 Elastic Load Balancing 運作狀態檢查，並指定要使用的可用區域和子網路

此範例更新指定的 Auto Scaling 群組，新增 Elastic Load Balancing 運作狀態檢查。此命令也使用多個可用區中的子網路 ID 清單更新 --vpc-zone-identifier 的值。

```
aws autoscaling update-auto-scaling-group \
```

```
--auto-scaling-group-name my-asg \  
--health-check-type ELB \  
--health-check-grace-period 600 \  
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Elastic Load Balancing](#) 和 [Amazon EC2 Auto Scaling](#)。

範例 3：更新置放群組和終止政策

此範例更新要使用的置放群組和終止政策。

```
aws autoscaling update-auto-scaling-group \  
--auto-scaling-group-name my-asg \  
--placement-group my-placement-group \  
--termination-policies "OldestInstance"
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Auto Scaling 群組](#)。

範例 4：使用最新版的啟動範本

此範例更新 Auto Scaling 群組，使用最新版的指定啟動範本。

```
aws autoscaling update-auto-scaling-group \  
--auto-scaling-group-name my-asg \  
--launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest'
```

此命令不會產生輸出。

如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的 [啟動範本](#)。

範例 5：使用特定版本的啟動範本

此範例更新指定的 Auto Scaling 群組，使用啟動範本的特定版本，而非最新或預設版本。

```
aws autoscaling update-auto-scaling-group \  

```

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

此命令不會產生輸出。

如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的[啟動範本](#)。

範例 6：定義混合執行個體政策，並啟用容量重新平衡

此範例更新指定的 Auto Scaling 群組，使用混合執行個體政策，並啟用容量重新平衡。此結構可讓您指定具有 Spot 和隨需容量的群組，並針對不同的架構使用不同的啟動範本。

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

config.json 的內容：

```
{  
  "AutoScalingGroupName": "my-asg",  
  "CapacityRebalance": true,  
  "MixedInstancesPolicy": {  
    "LaunchTemplate": {  
      "LaunchTemplateSpecification": {  
        "LaunchTemplateName": "my-launch-template-for-x86",  
        "Version": "$Latest"  
      },  
      "Overrides": [  
        {  
          "InstanceType": "c6g.large",  
          "LaunchTemplateSpecification": {  
            "LaunchTemplateName": "my-launch-template-for-arm",  
            "Version": "$Latest"  
          },  
          "Capacity": 1  
        },  
        {  
          "InstanceType": "c5.large",  
          "Capacity": 1  
        },  
        {  
          "InstanceType": "c5a.large",  
          "Capacity": 1  
        }  
      ]  
    }  
  },  
}
```

```
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    }
}
```

此命令不會產生輸出。

如需詳細資訊，請參閱「Amazon EC2 Auto Scaling 使用者指南」中的[具備多個執行個體類型及購買選項的 Auto Scaling 群組](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [UpdateAutoScalingGroup](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
    }
```

```

        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
    .waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group
" + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [UpdateAutoScalingGroup](#)。

Kotlin

適用於 Kotlin 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }
}

```

```

val groupRequest =
    UpdateAutoScalingGroupRequest {
        maxSize = 3
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
        autoScalingGroupName = groupName
        launchTemplate = templateSpecification
    }

val groupsRequestWaiter =
    DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
    autoScalingClient.updateAutoScalingGroup(groupRequest)
    autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
    println("You successfully updated the Auto Scaling group $groupName")
}
}

```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [UpdateAutoScalingGroup](#)。

PHP

適用於 PHP 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

public function updateAutoScalingGroup($autoScalingGroupName, $args)
{
    if (array_key_exists('MaxSize', $args)) {
        $maxSize = ['MaxSize' => $args['MaxSize']];
    } else {
        $maxSize = [];
    }
}

```



```
}  
if (array_key_exists('MinSize', $args)) {  
    $minSize = ['MinSize' => $args['MinSize']];  
} else {  
    $minSize = [];  
}  
$parameters = ['AutoScalingGroupName' => $autoScalingGroupName];  
$parameters = array_merge($parameters, $minSize, $maxSize);  
return $this->autoScalingClient->updateAutoScalingGroup($parameters);  
}
```

- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [UpdateAutoScalingGroup](#)。

PowerShell

Tools for PowerShell V4

範例 1：此範例更新指定的 Auto Scaling 群組的大小下限和上限。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

範例 2：此範例更新指定的 Auto Scaling 群組的預設冷卻時間。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

範例 3：此範例更新指定的 Auto Scaling 群組的可用區域。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-west-2a", "us-west-2b")
```

範例 4：此範例更新指定的 Auto Scaling 群組，使用 Elastic Load Balancing 運作狀態檢查。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -  
HealthCheckGracePeriod 60
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V4)》中的 [UpdateAutoScalingGroup](#)。

Tools for PowerShell V5

範例 1：此範例更新指定的 Auto Scaling 群組的大小下限和上限。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

範例 2：此範例更新指定的 Auto Scaling 群組的預設冷卻時間。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

範例 3：此範例更新指定的 Auto Scaling 群組的可用區域。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-west-2a", "us-west-2b")
```

範例 4：此範例更新指定的 Auto Scaling 群組，使用 Elastic Load Balancing 運作狀態檢查。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -  
HealthCheckGracePeriod 60
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考 (V5)》中的 [UpdateAutoScalingGroup](#)。

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
```

```

    """
    self.autoscaling_client = autoscaling_client

def update_group(self, group_name: str, **kwargs: Any) -> None:
    """
    Updates an Auto Scaling group.

    :param group_name: The name of the group to update.
    :param kwargs: Keyword arguments to pass through to the service.
    :return: None
    :raises ClientError: If there is an error updating the Auto Scaling
group.
    """
    try:
        self.autoscaling_client.update_auto_scaling_group(
            AutoScalingGroupName=group_name, **kwargs
        )
        logger.info(f"Successfully updated Auto Scaling group {group_name}.")

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        logger.error(f"Failed to update Auto Scaling group {group_name}.")
        if error_code == "ResourceInUse":
            logger.error(
                "The Auto Scaling group '%s' is currently in use and cannot
be modified. Please try again later.",
                group_name,
            )
        elif error_code == "ScalingActivityInProgress":
            logger.error(
                f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'."
                "Please wait for the activity to complete before attempting
to update the group."
            )
            logger.error(f"Full error:\n\t{err}")
            raise

```

- 如需 API 詳細資訊，請參閱《適用於 Python (Boto3) 的AWS SDK API 參考》中的 [UpdateAutoScalingGroup](#)。

Rust

適用於 Rust 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(),
Error> {
    client
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
        .send()
        .await?;

    println!("Updated AutoScaling group");

    Ok(())
}
```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的 AWS SDK API 參考》中的 [UpdateAutoScalingGroup](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_max_size = 5
```

```
TRY.  
    ao_asc->updateautoscalinggroup(  
        iv_autoscalinggroupname = iv_group_name  
        iv_maxsize = iv_max_size  
        iv_minsize = iv_min_size ).  
  
    MESSAGE 'Auto Scaling group updated successfully' TYPE 'I'.  
  
CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).  
    RAISE EXCEPTION lo_contention.  
CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).  
    RAISE EXCEPTION lo_activity_in_progress.  
CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).  
    RAISE EXCEPTION lo_generic_exception.  
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [UpdateAutoScalingGroup](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDKs 的 Auto Scaling 案例

下列程式碼範例示範如何在 Auto Scaling AWS SDKs 中實作常見案例。這些案例會向您展示如何呼叫 Auto Scaling 中的多個函數，或與其他 AWS 服務組合來完成特定任務。每個案例均包含完整原始碼的連結，您可在連結中找到如何設定和執行程式碼的相關指示。

案例的目標是獲得中等水平的經驗，協助您了解內容中的服務動作。

範例

- [使用 AWS SDK 建置和管理彈性服務](#)

使用 AWS SDK 建置和管理彈性服務

下列程式碼範例示範如何建立負載平衡的 Web 服務，以傳回書籍、影片和歌曲建議。此範例顯示服務如何回應失故障，以及如何在發生故障時重組服務以提高復原能力。

- 使用 Amazon EC2 Auto Scaling 群組根據啟動範本建立 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，並將執行個體數量保持在指定範圍內。
- 使用 Elastic Load Balancing 處理和分發 HTTP 請求。
- 監控 Auto Scaling 群組中執行個體的運作狀態，並且只將請求轉送給運作良好的執行個體。
- 在每個 EC2 執行個體上執行一個 Python Web 伺服器來處理 HTTP 請求。Web 伺服器會回應建議和運作狀態檢查。
- 使用 Amazon DynamoDB 資料表模擬建議服務。
- 透過更新 AWS Systems Manager 參數來控制 Web 伺服器對請求和運作狀態檢查的回應。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
static async Task Main(string[] args)
{
    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
        .Build();

    // Set up dependency injection for the AWS services.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft",
                    LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft",
                    LogLevel.Trace))
        .Build();
}
```

```

        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
                .AddAWSService<IAmazonDynamoDB>()
                .AddAWSService<IAmazonElasticLoadBalancingV2>()
                .AddAWSService<IAmazonSimpleSystemsManagement>()
                .AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalerWrapper>()
                .AddTransient<ElasticLoadBalancerWrapper>()
                .AddTransient<SmParameterWrapper>()
                .AddTransient<Recommendations>()
                .AddSingleton<IConfiguration>(_configuration)
        )
        .Build();

ServicesSetup(host);
ResourcesSetup();

try
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
    Console.WriteLine(new string('-', 80));
    await Deploy(true);

    Console.WriteLine("Now let's begin the scenario.");
    Console.WriteLine(new string('-', 80));
    await Demo(true);

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Finally, let's clean up our resources.");
    Console.WriteLine(new string('-', 80));

    await DestroyResources(true);

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
    Console.WriteLine(new string('-', 80));
}
catch (Exception ex)
{
    Console.WriteLine(new string('-', 80));

```

```

        Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
        await DestroyResources(true);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Setup any common resources, also used for integration testing.
/// </summary>
public static void ResourcesSetup()
{
    _httpClient = new HttpClient();
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
    _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    _recommendations = host.Services.GetRequiredService<Recommendations>();
    _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
    _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
}

/// <summary>
/// Deploy necessary resources for the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Deploy(bool interactive)
{
    var protocol = "HTTP";
    var port = 80;
    var sshPort = 22;

    Console.WriteLine(

```



```

        "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
        "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
        "against various kinds of failures.\n\n" +
        "Some of the resources create by this demo are:\n");

    Console.WriteLine(
        "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
    Console.WriteLine(
        "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
    Console.WriteLine(
        "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
    Console.WriteLine(
        "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
    if (interactive)
        Console.ReadLine();

    // Create and populate the DynamoDB table.
    var databaseTableName = _configuration["databaseName"];
    var recommendationsPath = Path.Join(_configuration["resourcePath"],
        "recommendations_objects.json");
    Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
    await _recommendations.CreateDatabaseWithName(databaseTableName);
    await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
    Console.WriteLine(new string('-', 80));

    // Create the EC2 Launch Template.

    Console.WriteLine(
        $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
        + "\nThis script starts a Python web server defined in the
'server.py' script. The web server\n"

```

```

        + "listens to HTTP requests on port 80 and responds to requests to
        '/' and to '/healthcheck'.\n"
        + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
        + "run a web server, such as Apache, with least-privileged
credentials.");
    Console.WriteLine(
        "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
        + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
        + "that control the flow of the demo.");

    var startupScriptPath = Path.Join(_configuration["resourcePath"],
        "server_startup_script.sh");
    var instancePolicyPath = Path.Join(_configuration["resourcePath"],
        "instance_policy.json");
    await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
    Console.WriteLine(new string('-', 80));

    Console.WriteLine(
        "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
        + "Availability Zone.\n");
    var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
    await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
    Console.WriteLine(new string('-', 80));

    Console.WriteLine(
        "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
        + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you're ready to continue.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("Creating variables that control the flow of the
demo.");
    await _smParameterWrapper.Reset();

```

```

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
            _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
            _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroup
protocol, port, defaultVpc.VpcId);

        await
            _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
subnetIds, targetGroup);
        await
            _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
            _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
var loadBalancerAccess = await
            _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
                _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
                _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
ipString);

```

```
var sshPortIsOpen =
_autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
ipString);

if (!portIsOpen)
{
    Console.WriteLine(
        "\nFor this example to work, the default security group for
your default VPC must\n"
        + "allows access from this computer. You can either add it
automatically from this\n"
        + "example or add it yourself using the AWS Management
Console.\n");

    if (!interactive || GetYesNoResponse(
        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
    {
        await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
    }
}

if (!sshPortIsOpen)
{
    if (!interactive || GetYesNoResponse(
        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
    {
        await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
    }
}

loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
}

if (loadBalancerAccess)
{
    Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
    Console.WriteLine($"http://{endPoint}\n");
}
```

```

    }
    else
    {
        Console.WriteLine(
            "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
            + "manually verifying that your VPC and security group are
configured correctly and that\n"
            + "you can successfully make a GET request to the load balancer
endpoint:\n");
        Console.WriteLine($"http://{endPoint}\n");
    }
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
    if (interactive)
        Console.ReadLine();
    return true;
}

/// <summary>
/// Demonstrate the steps of the scenario.
/// </summary>
/// <param name="interactive">True to run as an interactive scenario.</param>
/// <returns>Async task.</returns>
public static async Task<bool> Demo(bool interactive)
{
    var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
        "ssm_only_policy.json");

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Resetting parameters to starting values for demo.");
    await _smParameterWrapper.Reset();

    Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
        "to create situations where the web service fails, and
shows how using a resilient\n" +
        "architecture can keep the web service running in spite
of these failures.");
    Console.WriteLine(new string('-', 88));
    Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
    if (interactive)

```

```

        await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
            $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
            $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(

```

```

        _autoScalerWrapper.BadCredsPolicyName,
        _autoScalerWrapper.BadCredsRoleName,
        _autoScalerWrapper.BadCredsProfileName,
        ssmOnlyPolicy,
        new List<string> { "AmazonSSMManagedInstanceCore" }
    );
    var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
    var badInstanceId = instances.First();
    var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
    Console.WriteLine(
        $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
        "bad credentials...\n"
    );
    await _autoScalerWrapper.ReplaceInstanceProfile(
        badInstanceId,
        _autoScalerWrapper.BadCredsProfileName,
        instanceProfile.AssociationId
    );
    Console.WriteLine(
        "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
        "depending on which instance is selected by the load balancer.\n"
    );
    if (interactive)
        await DemoActionChoices();

    Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
    Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
    Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
    Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
    Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

    Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
    Console.WriteLine("and take that instance out of rotation.");

```

```
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
```



```

        Console.WriteLine($"\\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
                _elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
                _elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
                _autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
                _autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
                _autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName

```

```

        );
        await
        _recommendations.DestroyDatabaseByName(_recommendations.TableName);
    }
    else
    {
        Console.WriteLine(
            "Ok, we'll leave the resources intact.\n" +
            "Don't forget to delete them when you're done with them or you
            might incur unexpected charges."
        );
    }

    Console.WriteLine(new string('-', 80));
    return true;
}

```

建立包裝 Auto Scaling 和 Amazon EC2 動作的類別。

```

/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

```

```

public string GroupName => _groupName;
public string KeyPairName => _keyPairName;
public string LaunchTemplateName => _launchTemplateName;
public string InstancePolicyName => _instancePolicyName;
public string BadCredsProfileName => _badCredsProfileName;
public string BadCredsRoleName => _badCredsRoleName;
public string BadCredsPolicyName => _badCredsPolicyName;

/// <summary>
/// Constructor for the AutoScalerWrapper.
/// </summary>
/// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
/// <param name="amazonEc2">The injected EC2 client.</param>
/// <param name="amazonIam">The injected IAM client.</param>
/// <param name="amazonSsm">The injected SSM client.</param>
public AutoScalerWrapper(
    IAmazonAutoScaling amazonAutoScaling,
    IAmazonEC2 amazonEc2,
    IAmazonSimpleSystemsManagement amazonSsm,
    IAmazonIdentityManagementService amazonIam,
    IConfiguration configuration,
    ILogger<AutoScalerWrapper> logger)
{
    _amazonAutoScaling = amazonAutoScaling;
    _amazonEc2 = amazonEc2;
    _amazonSsm = amazonSsm;
    _amazonIam = amazonIam;
    _logger = logger;

    var prefix = configuration["resourcePrefix"];
    _instanceType = configuration["instanceType"];
    _amiParam = configuration["amiParam"];

    _launchTemplateName = prefix + "-template";
    _groupName = prefix + "-group";
    _instancePolicyName = prefix + "-pol";
    _instanceRoleName = prefix + "-role";
    _instanceProfileName = prefix + "-prof";
    _badCredsPolicyName = prefix + "-bc-pol";
    _badCredsRoleName = prefix + "-bc-role";
    _badCredsProfileName = prefix + "-bc-prof";
    _keyPairName = prefix + "-key-pair";
}

```

```

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
    a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance. The role has attached policies that specify the AWS permissions
    granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
    the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
            "\"Version\": \"2012-10-17\", " +
            "\"Statement\": [{" +
                "\"Effect\": \"Allow\", " +
                "\"Principal\": { " +
                "\"Service\": [ " +
                    "\"ec2.amazonaws.com\" " +
                "]" +
                "}, " +
                "\"Action\": \"sts:AssumeRole\" " +
            "}] " +
            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest

```

```
        {
            PolicyName = policyName,
            PolicyDocument = policyDocument
        });
        policyArn = createPolicyResult.Policy.Arn;
    }
    catch (EntityAlreadyExistsException)
    {
        // The policy already exists, so we look it up to get the Arn.
        var policiesPaginator = _amazonIam.Paginators.ListPolicies(
            new ListPoliciesRequest()
            {
                Scope = PolicyScopeType.Local
            });
        // Get the entire list using the paginator.
        await foreach (var policy in policiesPaginator.Policies)
        {
            if (policy.PolicyName.Equals(policyName))
            {
                policyArn = policy.Arn;
            }
        }

        if (policyArn == null)
        {
            throw new InvalidOperationException("Policy not found");
        }
    }

    try
    {
        await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = assumeRoleDoc,
        });
        await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
        {
            RoleName = roleName,
            PolicyArn = policyArn
        });
        if (awsManagedPolicies != null)
        {
            foreach (var awsPolicy in awsManagedPolicies)
```

```
        {
            await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
            {
                PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                RoleName = roleName
            });
        }
    }
}
catch (EntityAlreadyExistsException)
{
    Console.WriteLine("Role already exists.");
}

string profileArn = "";
try
{
    var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
    new CreateInstanceProfileRequest()
    {
        InstanceProfileName = profileName
    });
    // Allow time for the profile to be ready.
    profileArn = profileCreateResponse.InstanceProfile.Arn;
    Thread.Sleep(10000);
    await _amazonIam.AddRoleToInstanceProfileAsync(
        new AddRoleToInstanceProfileRequest()
        {
            InstanceProfileName = profileName,
            RoleName = roleName
        });
}
catch (EntityAlreadyExistsException)
{
    Console.WriteLine("Policy already exists.");
    var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
        new GetInstanceProfileRequest()
        {
            InstanceProfileName = profileName
        });
    profileArn = profileGetResponse.InstanceProfile.Arn;
```

```
    }
    return profileArn;
}

/// <summary>
/// Create a new key pair and save the file.
/// </summary>
/// <param name="newKeyPairName">The name of the new key pair.</param>
/// <returns>Async task.</returns>
public async Task CreateKeyPair(string newKeyPairName)
{
    try
    {
        var keyResponse = await _amazonEc2.CreateKeyPairAsync(
            new CreateKeyPairRequest() { KeyName = newKeyPairName });
        await File.WriteAllTextAsync($"{newKeyPairName}.pem",
            keyResponse.KeyPair.KeyMaterial);
        Console.WriteLine($"Created key pair {newKeyPairName}.");
    }
    catch (AlreadyExistsException)
    {
        Console.WriteLine("Key pair already exists.");
    }
}

/// <summary>
/// Delete the key pair and file by name.
/// </summary>
/// <param name="deleteKeyPairName">The key pair to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteKeyPairByName(string deleteKeyPairName)
{
    try
    {
        await _amazonEc2.DeleteKeyPairAsync(
            new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
        File.Delete($"{deleteKeyPairName}.pem");
    }
    catch (FileNotFoundException)
    {
        Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
    }
}
```

```

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
    Scaling.
    /// The launch template specifies a Bash script in its user data field that
    runs after
    /// the instance is started. This script installs the Python packages and
    starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
    run.</param>
    /// <param name="instancePolicyPath">The path to a permissions policy to
    create and attach to the profile.</param>
    /// <returns>The template object.</returns>
    public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
    startupScriptPath, string instancePolicyPath)
    {
        try
        {
            await CreateKeyPair(_keyPairName);
            await CreateInstanceProfileWithName(_instancePolicyName,
            _instanceRoleName,
                _instanceProfileName, instancePolicyPath);

            var startServerText = await File.ReadAllTextAsync(startupScriptPath);
            var plainTextBytes =
            System.Text.Encoding.UTF8.GetBytes(startServerText);

            var amiLatest = await _amazonSsm.GetParameterAsync(
                new GetParameterRequest() { Name = _amiParam });
            var amiId = amiLatest.Parameter.Value;
            var launchTemplateResponse = await
            _amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new
                                LaunchTemplateIamInstanceProfileSpecificationRequest()

```



```

        {
            Name = _instanceProfileName
        },
        KeyName = _keyPairName,
        UserData = System.Convert.ToBase64String(plainTextBytes)
    }
    });
    return launchTemplateResponse.LaunchTemplate;
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode ==
        "InvalidLaunchTemplateName.AlreadyExistsException")
    {
        _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
            $"Please try again with a unique name.");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
    throw;
}
}

/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    try
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z =>
            z.ZoneName).ToList();
    }
    catch (AmazonEC2Exception ec2Exception)

```

```

        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
size {groupSize}.");
        }
    }

```

```

        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>

```

```

    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
        }
    }

```

```

        throw;
    }
}

/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
                LaunchTemplateName = templateName
            });
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
            "InvalidLaunchTemplateName.NotFoundException")
        {
            _logger.LogError(
                $"Could not delete the template, the name
{ _launchTemplateName} was not found.");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
        throw;
    }
}

/// <summary>
/// Detaches a role from an instance profile, detaches policies from the
role,
/// and deletes all the resources.
/// </summary>

```

```
/// <param name="profileName">The name of the profile to delete.</param>
/// <param name="roleName">The name of the role to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteInstanceProfile(string profileName, string roleName)
{
    try
    {
        await _amazonIam.RemoveRoleFromInstanceProfileAsync(
            new RemoveRoleFromInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
                RoleName = roleName
            });
        await _amazonIam.DeleteInstanceProfileAsync(
            new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
        var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
            new ListAttachedRolePoliciesRequest() { RoleName = roleName });
        foreach (var policy in attachedPolicies.AttachedPolicies)
        {
            await _amazonIam.DetachRolePolicyAsync(
                new DetachRolePolicyRequest()
                {
                    RoleName = roleName,
                    PolicyArn = policy.PolicyArn
                });
            // Delete the custom policies only.
            if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
            {
                await _amazonIam.DeletePolicyAsync(
                    new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                    {
                        PolicyArn = policy.PolicyArn
                    });
            }
        }

        await _amazonIam.DeleteRoleAsync(
            new DeleteRoleRequest() { RoleName = roleName });
    }
    catch (NoSuchEntityException)
    {
        Console.WriteLine($"Instance profile {profileName} does not exist.");
    }
}
```

```

    }
}

/// <summary>
/// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
/// </summary>
/// <param name="group">The name of the auto scaling group.</param>
/// <returns>A collection of instance Ids.</returns>
public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
{
    var instanceResponse = await
        _amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
    var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
        g => g.Instances.Select(i => i.InstanceId));
    return instanceIds;
}

/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    try
    {
        var response = await
            _amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
        return response.IamInstanceProfileAssociations[0];
    }
    catch (AmazonEC2Exception ec2Exception)

```

```

        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
        }
    }

```



```

        // Allow time before resetting.
        Thread.Sleep(25000);

        await _amazonEc2.RebootInstancesAsync(
            new RebootInstancesRequest(new List<string>() { instanceId }));
        Thread.Sleep(25000);
        var instanceReady = false;
        var retries = 5;
        while (retries-- > 0 && !instanceReady)
        {
            var instancesPaginator =
                _amazonSsm.Paginators.DescribeInstanceInformation(
                    new DescribeInstanceInformationRequest());
            // Get the entire list using the paginator.
            await foreach (var instance in
instancesPaginator.InstanceInformationList)
            {
                instanceReady = instance.InstanceId == instanceId;
                if (instanceReady)
                {
                    break;
                }
            }
        }
        Console.WriteLine("Waiting for instance to be running.");
        await WaitForInstanceState(instanceId, InstanceStateName.Running);
        Console.WriteLine("Instance ready.");
        Console.WriteLine($"Sending restart command to instance
{instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                    {
                        "commands",
                        new List<string>() { "cd / && sudo python3 server.py
80" }
                    }
                }
            }
        ));

```

```

        Console.WriteLine($"Restarted the web server on instance
{instanceId}");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Try to terminate an instance by its Id.
/// </summary>
/// <param name="instanceId">The Id of the instance to terminate.</param>
/// <returns>Async task.</returns>
public async Task TryTerminateInstanceById(string instanceId)
{
    var stopping = false;
    Console.WriteLine($"Stopping {instanceId}...");
    while (!stopping)
    {
        try
        {
            await
            _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                new TerminateInstanceInAutoScalingGroupRequest()
                {
                    InstanceId = instanceId,
                    ShouldDecrementDesiredCapacity = false
                });
            stopping = true;
        }
        catch (ScalingActivityInProgressException)
    }
}

```

```

        {
            Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
/// waits and retries until the group is successfully deleted.
/// </summary>
/// <param name="groupName">The name of the group to try to delete.</param>
/// <returns>Async task.</returns>
public async Task TryDeleteGroupByName(string groupName)
{
    var stopped = false;
    while (!stopped)
    {
        try
        {
            await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                new DeleteAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName
                });
            stopped = true;
        }
        catch (Exception e)
            when ((e is ScalingActivityInProgressException)
                || (e is Amazon.AutoScaling.Model.ResourceInUseException))
        {
            Console.WriteLine($"Some instances are still running.
Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Terminate instances and delete the Auto Scaling group by name.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>

```

```

    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
        _amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }

    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {

```

```

        Filters = new List<Amazon.EC2.Model.Filter>()
        {
            new ("group-name", new List<string>() { "default" }),
            new ("vpc-id", new List<string>() { vpc.VpcId })
        }
    });
    return groupResponse.SecurityGroups[0];
}

/// <summary>
/// Verify the default security group of a Vpc allows ingress from the
calling computer.
/// This can be done by allowing ingress from this computer's IP address.
/// In some situations, such as connecting from a corporate network, you must
instead specify
/// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
/// If you do, be sure to remove public access when you're done.
/// </summary>
/// <param name="vpc">The group to check.</param>
/// <param name="port">The port to verify.</param>
/// <param name="ipAddress">This computer's IP address.</param>
/// <returns>True if the ip address is allowed on the group.</returns>
public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
{
    var portIsOpen = false;
    foreach (var ipPermission in group.IpPermissions)
    {
        if (ipPermission.FromPort == port)
        {
            foreach (var ipRange in ipPermission.Ipv4Ranges)
            {
                var cidr = ipRange.CidrIp;
                if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                {
                    portIsOpen = true;
                }
            }

            if (ipPermission.PrefixListIds.Any())
            {
                portIsOpen = true;
            }
        }
    }
}

```

```

        if (!portIsOpen)
        {
            Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
        }
        else
        {
            break;
        }
    }
}

return portIsOpen;
}

/// <summary>
/// Add an ingress rule to the specified security group that allows access on
the
/// specified port from the specified IP address.
/// </summary>
/// <param name="groupId">The Id of the security group to modify.</param>
/// <param name="port">The port to open.</param>
/// <param name="ipAddress">The IP address to allow access.</param>
/// <returns>Async task.</returns>
public async Task OpenInboundPort(string groupId, int port, string ipAddress)
{
    await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
        new AuthorizeSecurityGroupIngressRequest()
        {
            GroupId = groupId,
            IpPermissions = new List<IpPermission>()
            {
                new IpPermission()
                {
                    FromPort = port,
                    ToPort = port,
                    IpProtocol = "tcp",
                    Ipv4Ranges = new List<IpRange>()
                    {
                        new IpRange() { CidrIp = $"{ipAddress}/32" }
                    }
                }
            }
        }
    );
}

```

```

        }
    }
    });
}

/// <summary>
/// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
/// The
/// </summary>
/// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
/// <param name="targetGroupArn">The Arn for the target group.</param>
/// <returns>Async task.</returns>
public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
string targetGroupArn)
{
    await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
        new AttachLoadBalancerTargetGroupsRequest()
        {
            AutoScalingGroupName = autoScalingGroupName,
            TargetGroupARNs = new List<string>() { targetGroupArn }
        });
}

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

    // Wait until the instance is in the specified state.
    var hasState = false;
    do
    {
        // Wait 5 seconds.

```

```

        Thread.Sleep(5000);

        // Check for the desired state.
        var response = await _amazonEc2.DescribeInstancesAsync(request);
        var instance = response.Reservations[0].Instances[0];
        hasState = instance.State.Name == stateName;
        Console.WriteLine(". ");
    } while (!hasState);

    return hasState;
}
}

```

建立包裝 Elastic Load Balancing 動作的類別。

```

/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
    client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
    }
}

```



```

        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>

```

```

    public async Task<List<TargetHealthDescription>>
    CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
    how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
    times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
    unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>

```

```

    /// <param name="vpcId">The Id of the Vpc in which the load balancer
    exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
    ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
        _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
    subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
    List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
        _amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;
    }

```

```
// Wait for load balancer to be available.
var loadBalancerReady = false;
while (!loadBalancerReady)
{
    try
    {
        var describeResponse =
            await
            _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                new DescribeLoadBalancersRequest()
                {
                    Names = new List<string>() { name }
                });

        var loadBalancerState =
            describeResponse.LoadBalancers[0].State.Code;

        loadBalancerReady = loadBalancerState ==
            LoadBalancerStateEnum.Active;
    }
    catch (LoadBalancerNotFoundException)
    {
        loadBalancerReady = false;
    }
    Thread.Sleep(10000);
}
// Create the listener.
await _amazonElasticLoadBalancingV2.CreateListenerAsync(
    new CreateListenerRequest()
    {
        LoadBalancerArn = loadBalancerArn,
        Protocol = targetGroup.Protocol,
        Port = targetGroup.Port,
        DefaultActions = new List<Action>()
        {
            new Action()
            {
                Type = ActionTypeEnum.Forward,
                TargetGroupArn = targetGroup.TargetGroupArn
            }
        }
    });
return createLbResponse.LoadBalancers[0];
```

```

    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://{
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>

```

```

    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        }
                    );
            }
            catch { }
        }
    }

```

```

        });

        var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
        await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
            new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
        Console.WriteLine($"Deleted load balancing target group
{groupName}.");
        done = true;
    }
    catch (TargetGroupNotFoundException)
    {
        Console.WriteLine(
            $"Target group {groupName} not found, could not delete.");
        done = true;
    }
    catch (ResourceInUseException)
    {
        Console.WriteLine("Target group not yet released, waiting...");
        Thread.Sleep(10000);
    }
}
}
}
}

```

建立使用 DynamoDB 模擬建議服務的類別。

```

/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
/// movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>

```

```
/// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
/// <param name="configuration">The injected configuration.</param>
public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
{
    _amazonDynamoDb = amazonDynamoDb;
    _context = new DynamoDBContext(_amazonDynamoDb);
    _tableName = configuration["databaseName"]!;
}

/// <summary>
/// Create the DynamoDb table with a specified name.
/// </summary>
/// <param name="tableName">The name for the table.</param>
/// <returns>True when ready.</returns>
public async Task<bool> CreateDatabaseWithName(string tableName)
{
    try
    {
        Console.WriteLine($"Creating table {tableName}...");
        var createRequest = new CreateTableRequest()
        {
            TableName = tableName,
            AttributeDefinitions = new List<AttributeDefinition>()
            {
                new AttributeDefinition()
                {
                    AttributeName = "MediaType",
                    AttributeType = ScalarAttributeType.S
                },
                new AttributeDefinition()
                {
                    AttributeName = "ItemId",
                    AttributeType = ScalarAttributeType.N
                }
            },
            KeySchema = new List<KeySchemaElement>()
            {
                new KeySchemaElement()
                {
                    AttributeName = "MediaType",
                    KeyType = KeyType.HASH
                },
                new KeySchemaElement()
```



```
        {
            AttributeName = "ItemId",
            KeyType = KeyType.RANGE
        }
    },
    ProvisionedThroughput = new ProvisionedThroughput()
    {
        ReadCapacityUnits = 5,
        WriteCapacityUnits = 5
    }
};
await _amazonDynamoDb.CreateTableAsync(createRequest);

// Wait until the table is ACTIVE and then report success.
Console.WriteLine("\nWaiting for table to become active...");

var request = new DescribeTableRequest
{
    TableName = tableName
};

TableStatus status;
do
{
    Thread.Sleep(2000);

    var describeTableResponse = await
        _amazonDynamoDb.DescribeTableAsync(request);
    status = describeTableResponse.Table.TableStatus;

    Console.WriteLine(".");
}
while (status != "ACTIVE");

return status == TableStatus.ACTIVE;
}
catch (ResourceInUseException)
{
    Console.WriteLine($"Table {tableName} already exists.");
    return false;
}
}

/// <summary>
```

```

    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
recommendationsPath)
    {
        var recommendationsText = await
File.ReadAllTextAsync(recommendationsPath);
        var records =

JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}

```

建立包裝 Systems Manager 動作的類別。

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
    parameters
/// to drive the demonstration of resilient architecture, such as failure of a
    dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
        _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
```

```
/// Reset the Systems Manager parameters to starting values for the demo.
/// </summary>
/// <returns>Async task.</returns>
public async Task Reset()
{
    await this.PutParameterByName(_tableParameter, _tableName);
    await this.PutParameterByName(_failureResponseParameter, "none");
    await this.PutParameterByName(_healthCheckParameter, "shallow");
}

/// <summary>
/// Set the value of a named Systems Manager parameter.
/// </summary>
/// <param name="name">The name of the parameter.</param>
/// <param name="value">The value to set.</param>
/// <returns>Async task.</returns>
public async Task PutParameterByName(string name, string value)
{
    await _amazonSimpleSystemsManagement.PutParameterAsync(
        new PutParameterRequest() { Name = name, Value = value, Overwrite =
true });
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的下列主題。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)

- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
public class Main {  
  
    public static final String fileName = "C:\\\\AWS\\\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\\\AWS\\\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
    public static final String policyFile = "C:\\\\AWS\\\\resworkflow\\  
\\instance_policy.json"; // Modify file location.  
}
```

```

    public static final String ssmJSON = "C:\\\\AWS\\\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");

    public static void main(String[] args) throws IOException,
InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);

```

```

        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
            This concludes the demo of how to build and manage a resilient
service.

            To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
            that were created for this demo.
            """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                Okay, we'll leave the resources intact.
                Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
        throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);

```

```

        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
    InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""

```


Creating an EC2 launch template that runs '{startup_script}' when an instance starts.

This script starts a Python web server defined in the `server.py` script. The web server

listens to HTTP requests on port 80 and responds to requests to '/' and to '/healthcheck'.

For demo purposes, this server is run as the root user. In production, the best practice is to

run a web server, such as Apache, with least-privileged credentials.

The template also defines an IAM policy that each instance uses to assume a role that grants

permissions to access the DynamoDB recommendation table and Systems Manager parameters

that control the flow of the demo.

```
""");
```

```
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println(
```

```
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
```

```
System.out.println("*** Wait 30 secs for the VPC to be created");
```

```
TimeUnit.SECONDS.sleep(30);
```

```
AutoScaler autoScaler = new AutoScaler();
```

```
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
System.out.println("""
```

```
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
```

```
    HTTP requests. You can see these instances in the console or
continue with the demo.
```

```
    Press Enter when you're ready to continue.
```

```
""");
```

```
in.nextLine();
```

```
System.out.println(DASHES);
```

```

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load
balancer. The target group
            defines how the load balancer connects to instances. The load
balancer provides a
            single endpoint where clients connect and dispatches requests to
instances in the group.
            """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessful =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessful) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.

```

```

        String ipAddress =
        IOUtils.toString(response.getEntity().getContent(),
        StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
        ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group
for your default VPC must
                allow access from this computer. You can either add
it automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");

```

```

        System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
        System.out.println("you can successfully make a GET request to the
load balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and
shows how using a resilient
                architecture can keep the web service running in spite
of these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                """);
    demoChoices(loadBalancer);

    System.out.println(
        """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                """);

```

```

paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    ""
        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
    "");
demoChoices(loadBalancer);

System.out.println(
    ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
    "");
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns
a static response.
    The service still reports as healthy because health checks are
still shallow.
    """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance
id value.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);

```

```

        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
            + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
            """"
                Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                depending on which instance is selected by the load
balancer.

            """);

        demoChoices(loadBalancer);

        System.out.println("""
            Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
            This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
            """);

        System.out.println("""
            By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
            """);

        paramHelper.put(paramHelper.healthCheck, "deep");

```

```
        System.out.println("""
            Now, checking target health indicates that the instance with bad
credentials
            is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
            instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
            the load balancer takes unhealthy instances out of its rotation.
            """);

        demoChoices(loadBalancer);

        System.out.println(
            """
                Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start
a new instance to replace it.
                """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load
balancing rotation.
            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance
is running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
```

```

    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
        InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClientClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(

```



```

        new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load
balancer targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
                                Note that it can take a minute or two for the
health check to update
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {

```

```

        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

建立包裝 Auto Scaling 和 Amazon EC2 動作的類別。

```

public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {

```

```

        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile
is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)

```

```

        throws InterruptedException {
            // Create an IAM instance profile specification.
            software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                                // name.
                .build();

            // Replace the IAM instance profile association for the EC2 instance.
            ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

            try {
                getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
                // Handle the response as needed.
            } catch (Ec2Exception e) {
                // Handle exceptions, log, or report the error.
                System.err.println("Error: " + e.getMessage());
            }

            System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
            TimeUnit.SECONDS.sleep(15);
            boolean instReady = false;
            int tries = 0;

            // Reboot after 60 seconds
            while (!instReady) {
                if (tries % 6 == 0) {
                    getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                    System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
                }
                tries++;
            }

```

```

        try {
            TimeUnit.SECONDS.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
            Collections.singletonList("cd / && sudo python3 server.py
80")))
        .build();

    getSSMClient().sendCommand(sendCommandRequest);
    System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress)
{
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

```

```
/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        getInstanceProfileRequest =
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

        GetInstanceProfileResponse response =
        getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.instanceProfile().instanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
        RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
        DeleteInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
        rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
```

```

        DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(attachedPolicy.policyArn())
            .build();

        getIAMClient().detachRolePolicy(request);
        System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
    }

    getIAMClient().deleteRole(deleteRoleRequest);
    System.out.println("Instance profile and role deleted.");

} catch (IamException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

    getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network,
you

```

```

    * must instead specify a prefix list ID. You can also temporarily open the
port
    * to
    * any IP address while running this example. If you do, be sure to remove
    * public
    * access when you're done.
    *
    */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " +
secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " +
ipPermission);

                    for (IpRange ipRange : ipPermission.ipRanges()) {

```



```

        String cidrIp = ipRange.cidrIp();
        if (cidrIp.startsWith(ipAddress) ||
        cidrIp.equals("0.0.0.0/0")) {
            System.out.println(cidrIp + " is applicable");
            portIsOpen = true;
        }
    }

    if (!ipPermission.prefixListIds().isEmpty()) {
        System.out.println("Prefix lList is applicable");
        portIsOpen = true;
    }

    if (!portIsOpen) {
        System.out
            .println("The inbound rule does not appear to
be open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
    } else {
        break;
    }
}
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {

```

```

        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
    .autoScalingGroupName(asGroupName)
    .targetGroupARNs(targetGroupARN)
    .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
    System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

    String availabilityZones = String.join(",", availabilityZoneNames);
    LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

    String[] zones = availabilityZones.split(",");

```

```

        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability
Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")

```

```

        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}
}

```

```

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to
the policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
        .builder()
        .policyArn(policy.arn())
        .build();
        ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
            .listEntitiesForPolicy(listEntitiesRequest);
        if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
            || !listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()

```

```

        .policyArn(policy.arn())
        .roleName(roleName) // Specify the name of the IAM
role
        .build();

        getIAMClient().detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
        .policyArn(policy.arn())
        .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
}

// Delete the instance profile after removing all roles

```

```

        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

建立包裝 Elastic Load Balancing 動作的類別。

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()

            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())

```

```

        .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

                .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                .build();

            getLoadBalancerClient().deleteLoadBalancer(
                builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter

                .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {

```



```

        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load
    balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }
    }

```

```

        System.out.println("Status.." + success);
        return success;
    }

    /**
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
     instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
            .healthCheckPath("/healthcheck")
            .healthCheckTimeoutSeconds(5)
            .port(port)
            .vpcId(vpcId)
            .name(targetGroupName)
            .protocol(protocol)
            .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()

```

```
        .map(Subnet::subnetId)
        .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
        .subnets(subnetIdStrings)
        .name(lbName)
        .scheme("internet-facing")
        .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

        CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

        .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
```

```

        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
        + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

建立使用 DynamoDB 模擬建議服務的類別。

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder()
                    .tableName(tableName)

```

```

        .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " +
e.getMessage());
    }
    return false;
}

/*
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended,
such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException
{
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")

```

```

        .keyType(KeyType.HASH)
        .build(),
        KeySchemaElement.builder()
            .attributeName("ItemId")
            .keyType(KeyType.RANGE)
            .build())
    .provisionedThroughput(
        ProvisionedThroughput.builder()
            .readCapacityUnits(5L)
            .writeCapacityUnits(5L)
            .build())
    .build();

getDynamoDbClient().createTable(createTableRequest);
System.out.println("Creating table " + tableName + "...");

// Wait until the Amazon DynamoDB table is created.
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Table " + tableName + " created.");

// Add records to the table.
populateTable(fileName, tableName);
}
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws
IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();

```

```

        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
            TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}

```

建立包裝 Systems Manager 動作的類別。

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }
}

```

```
public void put(String name, String value) {
    SsmClient ssmClient = SsmClient.builder()
        .region(Region.US_EAST_1)
        .build();

    PutParameterRequest parameterRequest = PutParameterRequest.builder()
        .name(name)
        .value(value)
        .overwrite(true)
        .type("String")
        .build();

    ssmClient.putParameter(parameterRequest);
    System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)

- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

JavaScript

適用於 JavaScript (v3) 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "../steps-deploy.js";
```

```

import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 * class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}

```

建立步驟以部署所有資源。

```

import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

```

```
import {
    BatchWriteItemCommand,
    CreateTableCommand,
    DynamoDBClient,
    waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
    EC2Client,
    CreateKeyPairCommand,
    CreateLaunchTemplateCommand,
    DescribeAvailabilityZonesCommand,
    DescribeVpcsCommand,
    DescribeSubnetsCommand,
    DescribeSecurityGroupsCommand,
    AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
    IAMClient,
    CreatePolicyCommand,
    CreateRoleCommand,
    CreateInstanceProfileCommand,
    AddRoleToInstanceProfileCommand,
    AttachRolePolicyCommand,
    waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
    CreateAutoScalingGroupCommand,
    AutoScalingClient,
    AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
    CreateListenerCommand,
    CreateLoadBalancerCommand,
    CreateTargetGroupCommand,
    ElasticLoadBalancingV2Client,
    waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
    ScenarioOutput,
    ScenarioInput,
    ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```

import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",

```

```

        KeyType: "HASH",
      },
      {
        AttributeName: "ItemId",
        KeyType: "RANGE",
      },
    ],
  )),
);
await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
}),
new ScenarioOutput(
  "createdTable",
  MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioOutput(
  "populatingTable",
  MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioAction("populateTable", () => {
  const client = new DynamoDBClient({});
  /**
   * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
  */
  const recommendations = JSON.parse(
    readFileSync(join(RESOURCES_PATH, "recommendations.json")),
  );

  return client.send(
    new BatchWriteItemCommand({
      RequestItems: {
        [NAMES.tableName]: recommendations.map((item) => ({
          PutRequest: { Item: item },
        })),
      },
    }),
  );
}),
new ScenarioOutput(
  "populatedTable",
  MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioOutput(

```

```

    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(

```

```

    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
    ),
),
new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
        new CreateRoleCommand({
            RoleName: NAMES.instanceRoleName,
            AssumeRolePolicyDocument: readFileSync(
                join(ROOT, "assume-role-policy.json"),
            ),
        }),
    );
}),
new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
    ),
),
new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
),
new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
        new AttachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: state.instancePolicyArn,
        }),
    );
}),
new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),

```

```

    ),
    new ScenarioOutput(
      "creatingInstanceProfile",
      MESSAGES.creatingInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      ),
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
),

```



```

new ScenarioOutput(
  "addedRoleToInstanceProfile",
  MESSAGES.addedRoleToInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
...initParamsSteps,
new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
new ScenarioAction("createLaunchTemplate", async () => {
  const ssmClient = new SSMClient({});
  const { Parameter } = await ssmClient.send(
    new GetParameterCommand({
      Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
    }),
  );
  const ec2Client = new EC2Client({});
  await ec2Client.send(
    new CreateLaunchTemplateCommand({
      LaunchTemplateName: NAMES.launchTemplateName,
      LaunchTemplateData: {
        InstanceType: "t3.micro",
        ImageId: Parameter.Value,
        IamInstanceProfile: { Name: NAMES.instanceProfileName },
        UserData: readFileSync(
          join(RESOURCES_PATH, "server_startup_script.sh"),
        ).toString("base64"),
        KeyName: NAMES.keyPairName,
      },
    }),
  );
}),
new ScenarioOutput(
  "createdLaunchTemplate",
  MESSAGES.createdLaunchTemplate.replace(
    "${LAUNCH_TEMPLATE_NAME}",
    NAMES.launchTemplateName,
  ),
),
new ScenarioOutput(
  "creatingAutoScalingGroup",
  MESSAGES.creatingAutoScalingGroup.replace(
    "${AUTO_SCALING_GROUP_NAME}",
    NAMES.autoScalingGroupName,
  ),
),

```

```

    ),
    new ScenarioAction("createAutoScalingGroup", async (state) => {
        const ec2Client = new EC2Client({});
        const { AvailabilityZones } = await ec2Client.send(
            new DescribeAvailabilityZonesCommand({}),
        );
        state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
        const autoScalingClient = new AutoScalingClient({});
        await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
            autoScalingClient.send(
                new CreateAutoScalingGroupCommand({
                    AvailabilityZones: state.availabilityZoneNames,
                    AutoScalingGroupName: NAMES.autoScalingGroupName,
                    LaunchTemplate: {
                        LaunchTemplateName: NAMES.launchTemplateName,
                        Version: "$Default",
                    },
                    MinSize: 3,
                    MaxSize: 3,
                }),
            ),
        );
    }),
    new ScenarioOutput(
        "createdAutoScalingGroup",
        /**
         * @param {{ availabilityZoneNames: string[] }} state
         */
        (state) =>
            MESSAGES.createdAutoScalingGroup
                .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
                .replace(
                    "${AVAILABILITY_ZONE_NAMES}",
                    state.availabilityZoneNames.join(", "),
                ),
    ),
    new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
        type: "confirm",
    }),
    new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
    new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
    new ScenarioAction("getVpc", async (state) => {
        const client = new EC2Client({});
        const { Vpcs } = await client.send(

```

```

        new DescribeVpcsCommand({
            Filters: [{ Name: "is-default", Values: ["true"] }],
        }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
        new DescribeSubnetsCommand({
            Filters: [
                { Name: "vpc-id", Values: [state.defaultVpc] },
                { Name: "availability-zone", Values: state.availabilityZoneNames },
                { Name: "default-for-az", Values: ["true"] },
            ],
        }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
}),
new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
        MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
),
new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
    ),
),
new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
        new CreateTargetGroupCommand({
            Name: NAMES.loadBalancerTargetGroupName,
            Protocol: "HTTP",

```

```

        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
    )),
);
const targetGroup = TargetGroups[0];
state.targetGroupArn = targetGroup.TargetGroupArn;
state.targetGroupProtocol = targetGroup.Protocol;
state.targetGroupPort = targetGroup.Port;
}),
new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
    ),
),
new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
),
new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
        new CreateLoadBalancerCommand({
            Name: NAMES.loadBalancerName,
            Subnets: state.subnets,
        }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
        { client },
        { Names: [NAMES.loadBalancerName] },
    );
}),
new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
        .replace("${LB_NAME}", NAMES.loadBalancerName)
        .replace("${DNS_NAME}", state.loadBalancerDns),

```

```

    ),
    new ScenarioOutput(
      "creatingListener",
      MESSAGES.creatingLoadBalancerListener
        .replace("${LB_NAME}", NAMES.loadBalancerName)
        .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
    ),
    new ScenarioAction("createListener", async (state) => {
      const client = new ElasticLoadBalancingV2Client({});
      const { Listeners } = await client.send(
        new CreateListenerCommand({
          LoadBalancerArn: state.loadBalancerArn,
          Protocol: state.targetGroupProtocol,
          Port: state.targetGroupPort,
          DefaultActions: [
            { Type: "forward", TargetGroupArn: state.targetGroupArn },
          ],
        }),
      );
      const listener = Listeners[0];
      state.loadBalancerListenerArn = listener.ListenerArn;
    }),
    new ScenarioOutput("createdListener", (state) =>
      MESSAGES.createdLoadBalancerListener.replace(
        "${LB_LISTENER_ARN}",
        state.loadBalancerListenerArn,
      ),
    ),
    new ScenarioOutput(
      "attachingLoadBalancerTargetGroup",
      MESSAGES.attachingLoadBalancerTargetGroup
        .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
    ),
    new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new AttachLoadBalancerTargetGroupsCommand({
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          TargetGroupARNs: [state.targetGroupArn],
        }),
      );
    }),
    new ScenarioOutput(

```

```

    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**

```

```

    * @param {{ myIpRules: any[] }} state
    */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",

```

```

    })),
  );
},
),
new ScenarioOutput("addedInboundRule", (state) => {
  if (state.shouldAddInboundRule) {
    return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
  }
  return false;
})),
new ScenarioOutput("verifyingEndpoint", (state) =>
  MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
),
new ScenarioAction("verifyEndpoint", async (state) => {
  try {
    const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
      axios.get(`http://${state.loadBalancerDns}`),
    );
    state.endpointResponse = JSON.stringify(response.data, null, 2);
  } catch (e) {
    state.verifyEndpointError = e;
  }
})),
new ScenarioOutput("verifiedEndpoint", (state) => {
  if (state.verifyEndpointError) {
    console.error(state.verifyEndpointError);
  } else {
    return MESSAGES.verifiedEndpoint.replace(
      "${ENDPOINT_RESPONSE}",
      state.endpointResponse,
    );
  }
})),
saveState,
];

```

建立步驟以執行示範。

```

import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

```



```
import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
```

```

    "getRecommendation",
    async (state) => {
        const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
        if (loadBalancer) {
            state.loadBalancerDnsName = loadBalancer.DNSName;
            try {
                state.recommendation = (
                    await axios.get(`http://${state.loadBalancerDnsName}`)
                ).data;
            } catch (e) {
                state.recommendation = e instanceof Error ? e.message : e;
            }
        } else {
            throw new Error(MESSAGES.demoFindLoadBalancerError);
        }
    },
);

const getRecommendationResult = new ScenarioOutput(
    "getRecommendationResult",
    (state) =>
        `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
    { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
            Names: [NAMES.loadBalancerTargetGroupName],
        }),
    );
});

const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
        TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
);
state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
    "getHealthCheckResult",
    /**

```

```

    * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
    balancing-v2').TargetHealthDescription[]}} state
    */
    (state) => {
      const status = state.targetHealthDescriptions
        .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
        .join("\n");
      return `Health check:\n${status}`;
    },
    { preformatted: true },
  );

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

```

```

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
  ),
];

```

```

    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**

```

```

    * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
    */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(

```

```

        (info) => info.InstanceId === state.targetInstance.InstanceId,
    );

    if (!instance) {
        throw new Error("Instance not found.");
    }
});

await ssmClient.send(
    new SendCommandCommand({
        InstanceIds: [state.targetInstance.InstanceId],
        DocumentName: "AWS-RunShellScript",
        Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
    }),
);
},
),
new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}} state
     */
    (state) =>
        MESSAGES.demoTestBadCredentials.replace(
            "${INSTANCE_ID}",
            state.targetInstance.InstanceId,
        ),
),
loadBalancerLoop,
new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
),
new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
        process.exit();
    }
})),
new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
        new PutParameterCommand({

```

```

        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
    )),
    );
  )),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
    ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  )),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
    ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
),

```



```

new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
  type: "confirm",
}),
new ScenarioAction("failOpenExit", (state) => {
  if (!state.failOpenConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("failOpen", () => {
  const client = new SSMClient({});
  return client.send(
    new PutParameterCommand({
      Name: NAMES.ssmTableNameKey,
      Value: `fake-table-${Date.now()}`,
      Overwrite: true,
      Type: "String",
    }),
  );
}),
new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput(
  "resetTableConfirmation",
  MESSAGES.demoResetTableConfirmation,
  { type: "confirm" },
),
new ScenarioAction("resetTableExit", (state) => {
  if (!state.resetTableConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("resetTable", async () => {
  const client = new SSMClient({});
  await client.send(
    new PutParameterCommand({
      Name: NAMES.ssmTableNameKey,
      Value: NAMES.tableName,
      Overwrite: true,
      Type: "String",
    }),
  ),

```

```

    );
  }},
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
}

```

```

const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
await iamClient.send(
  new AddRoleToInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    RoleName: NAMES.ssmOnlyRoleName,
  }),
);

return InstanceProfile;
}

```

建立步驟以銷毀所有資源。

```

import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,

```

```

    TerminateInstanceInAutoScalingGroupCommand,
    UpdateAutoScalingGroupCommand,
    paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
    DeleteLoadBalancerCommand,
    DeleteTargetGroupCommand,
    DescribeTargetGroupsCommand,
    ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
    ScenarioOutput,
    ScenarioInput,
    ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "../constants.js";
import { findLoadBalancer } from "../shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
    loadState,
    new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
    new ScenarioAction(
        "abort",
        (state) => state.destroy === false && process.exit(),
    ),
    new ScenarioAction("deleteTable", async (c) => {
        try {
            const client = new DynamoDBClient({});
            await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
        } catch (e) {
            c.deleteTableError = e;
        }
    }),
    new ScenarioOutput("deleteTableResult", (state) => {
        if (state.deleteTableError) {
            console.error(state.deleteTableError);
            return MESSAGES.deleteTableError.replace(

```

```

        "${TABLE_NAME}",
        NAMES.tableName,
    );
}
return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
}),
new ScenarioAction("deleteKeyPair", async (state) => {
    try {
        const client = new EC2Client({});
        await client.send(
            new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
        );
        unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
        state.deleteKeyPairError = e;
    }
}),
new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
        console.error(state.deleteKeyPairError);
        return MESSAGES.deleteKeyPairError.replace(
            "${KEY_PAIR_NAME}",
            NAMES.keyPairName,
        );
    }
    return MESSAGES.deletedKeyPair.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
    );
}),
new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
        const client = new IAMClient({});
        const policy = await findPolicy(NAMES.instancePolicyName);

        if (!policy) {
            state.detachPolicyFromRoleError = new Error(
                `Policy ${NAMES.instancePolicyName} not found.`
            );
        } else {
            await client.send(
                new DetachRolePolicyCommand({
                    RoleName: NAMES.instanceRoleName,
                    PolicyArn: policy.Arn,
                })
            );
        }
    }
});

```

```

    })),
    );
  }
} catch (e) {
  state.detachPolicyFromRoleError = e;
}
})),
new ScenarioOutput("detachedPolicyFromRole", (state) => {
  if (state.detachPolicyFromRoleError) {
    console.error(state.detachPolicyFromRoleError);
    return MESSAGES.detachPolicyFromRoleError
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }
  return MESSAGES.detachedPolicyFromRole
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
})),
new ScenarioAction("deleteInstancePolicy", async (state) => {
  const client = new IAMClient({});
  const policy = await findPolicy(NAMES.instancePolicyName);

  if (!policy) {
    state.deletePolicyError = new Error(
      `Policy ${NAMES.instancePolicyName} not found.`
    );
  } else {
    return client.send(
      new DeletePolicyCommand({
        PolicyArn: policy.Arn,
      }),
    );
  }
})),
new ScenarioOutput("deletePolicyResult", (state) => {
  if (state.deletePolicyError) {
    console.error(state.deletePolicyError);
    return MESSAGES.deletePolicyError.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }
  return MESSAGES.deletedPolicy.replace(
    "${INSTANCE_POLICY_NAME}",

```

```

        NAMES.instancePolicyName,
    );
  }},
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }},
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }},
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }},
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",

```

```

        NAMES.instanceRoleName,
    );
}
return MESSAGES.deletedInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
);
}),
new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
        const client = new IAMClient({});
        await client.send(
            new DeleteInstanceProfileCommand({
                InstanceProfileName: NAMES.instanceProfileName,
            }),
        );
    } catch (e) {
        state.deleteInstanceProfileError = e;
    }
}),
new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
        console.error(state.deleteInstanceProfileError);
        return MESSAGES.deleteInstanceProfileError.replace(
            "${INSTANCE_PROFILE_NAME}",
            NAMES.instanceProfileName,
        );
    }
    return MESSAGES.deletedInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
    );
}),
new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
        await terminateGroupInstances(NAMES.autoScalingGroupName);
        await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
            await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
        });
    } catch (e) {
        state.deleteAutoScalingGroupError = e;
    }
}),
new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {

```



```

    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  })),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  })),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  })),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,

```

```

    })),
  );
  await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
    const lb = await findLoadBalancer(NAMES.loadBalancerName);
    if (lb) {
      throw new Error("Load balancer still exists.");
    }
  });
} catch (e) {
  state.deleteLoadBalancerError = e;
}
}),
new ScenarioOutput("deleteLoadBalancerResult", (state) => {
  if (state.deleteLoadBalancerError) {
    console.error(state.deleteLoadBalancerError);
    return MESSAGES.deleteLoadBalancerError.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }
  return MESSAGES.deletedLoadBalancer.replace(
    "${LB_NAME}",
    NAMES.loadBalancerName,
  );
}),
new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  try {
    const { TargetGroups } = await client.send(
      new DescribeTargetGroupsCommand({
        Names: [NAMES.loadBalancerTargetGroupName],
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      client.send(
        new DeleteTargetGroupCommand({
          TargetGroupArn: TargetGroups[0].TargetGroupArn,
        }),
      );
  } catch (e) {
    state.deleteLoadBalancerTargetGroupError = e;
  }
}

```

```

    }},
    new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
      if (state.deleteLoadBalancerTargetGroupError) {
        console.error(state.deleteLoadBalancerTargetGroupError);
        return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
          "${TARGET_GROUP_NAME}",
          NAMES.loadBalancerTargetGroupName,
        );
      }
      return MESSAGES.deletedLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }),
    new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
      try {
        const client = new IAMClient({});
        await client.send(
          new RemoveRoleFromInstanceProfileCommand({
            InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
            RoleName: NAMES.ssmOnlyRoleName,
          }),
        );
      } catch (e) {
        state.detachSsmOnlyRoleFromProfileError = e;
      }
    }),
    new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
      if (state.detachSsmOnlyRoleFromProfileError) {
        console.error(state.detachSsmOnlyRoleFromProfileError);
        return MESSAGES.detachSsmOnlyRoleFromProfileError
          .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
          .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
      }
      return MESSAGES.detachedSsmOnlyRoleFromProfile
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }),
    new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
      try {
        const iamClient = new IAMClient({});
        const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
        await iamClient.send(
          new DetachRolePolicyCommand({

```

```

        RoleName: NAMES.ssmOnlyRoleName,
        PolicyArn: ssmOnlyPolicy.Arn,
    )),
    );
} catch (e) {
    state.detachSsmOnlyCustomRolePolicyError = e;
}
}),
new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
        console.error(state.detachSsmOnlyCustomRolePolicyError);
        return MESSAGES.detachSsmOnlyCustomRolePolicyError
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
}),
new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
        const iamClient = new IAMClient({});
        await iamClient.send(
            new DetachRolePolicyCommand({
                RoleName: NAMES.ssmOnlyRoleName,
                PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
            )),
        );
    } catch (e) {
        state.detachSsmOnlyAWSRolePolicyError = e;
    }
}),
new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
        console.error(state.detachSsmOnlyAWSRolePolicyError);
        return MESSAGES.detachSsmOnlyAWSRolePolicyError
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
}),
new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {

```

```

    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  })),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  })),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  })),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
  }));

```

```

    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
    );
  })),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  })),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  })),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
      state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,

```

```

        FromPort: 80,
        ToPort: 80,
        IpProtocol: "tcp",
    )),
    );
} catch (e) {
    state.revokeSecurityGroupIngressError = e;
}
},
),
new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
        console.error(state.revokeSecurityGroupIngressError);
        return MESSAGES.revokeSecurityGroupIngressError.replace(
            "${IP}",
            state.myIp,
        );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
}),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
    const client = new IAMClient({});
    const paginatedPolicies = paginateListPolicies({ client }, {});
    for await (const page of paginatedPolicies) {
        const policy = page.Policies.find((p) => p.PolicyName === policyName);
        if (policy) {
            return policy;
        }
    }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
    const client = new AutoScalingClient({});
    try {
        await client.send(
            new DeleteAutoScalingGroupCommand({

```

```

        AutoScalingGroupName: groupName,
      )),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    )),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        )),
      ),
  );
}
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {

```



```
        return group;
    }
}
throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的下列主題。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Python

適用於 Python 的 SDK (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
    resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
        such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
```

```
self.ssh_port = 22

prefix = "doc-example-resilience"
self.target_group_name = f"{prefix}-tg"
self.load_balancer_name = f"{prefix}-lb"

def deploy(self) -> None:
    """
    Deploys the resources required for the resilient service, including the
    DynamoDB table,
    EC2 instances, Auto Scaling group, and load balancer.
    """
    recommendations_path = f"{self.resource_path}/recommendations.json"
    startup_script = f"{self.resource_path}/server_startup_script.sh"
    instance_policy = f"{self.resource_path}/instance_policy.json"

    logging.info("Starting deployment of resources for the resilient
    service.")

    logging.info(
        "Creating and populating DynamoDB table '%s'.",
        self.recommendation.table_name,
    )
    self.recommendation.create()
    self.recommendation.populate(recommendations_path)

    logging.info(
        "Creating an EC2 launch template with the startup script '%s'.",
        startup_script,
    )
    self.autoscaler.create_template(startup_script, instance_policy)

    logging.info(
        "Creating an EC2 Auto Scaling group across multiple Availability
    Zones."
    )
    zones = self.autoscaler.create_autoscaling_group(3)

    logging.info("Creating variables that control the flow of the demo.")
    self.param_helper.reset()

    logging.info("Creating Elastic Load Balancing target group and load
    balancer.")
```

```

vpc = self.autoscaler.get_default_vpc()
subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
target_group = self.loadbalancer.create_target_group(
    self.target_group_name, self.protocol, self.port, vpc["VpcId"]
)
self.loadbalancer.create_load_balancer(
    self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
)
self.loadbalancer.create_listener(self.load_balancer_name, target_group)

self.autoscaler.attach_load_balancer_target_group(target_group)

logging.info("Verifying access to the load balancer endpoint.")
endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

if not lb_success:
    logging.warning(
        "Couldn't connect to the load balancer. Verifying that the port
is open..."
    )
    sec_group, port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.port, current_ip_address
    )
    sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.ssh_port, current_ip_address
    )
    if not port_is_open:
        logging.warning(
            "The default security group for your VPC must allow access
from this computer."
        )
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
            f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
            q.is_yesno,
        ):
            self.autoscaler.open_inbound_port(
                sec_group["GroupId"], self.port, current_ip_address
            )

```

```

        if not ssh_port_is_open:
            if q.ask(
                f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                q.is_yesno,
            ):
                self.autoscaler.open_inbound_port(
                    sec_group["GroupId"], self.ssh_port, current_ip_address
                )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")

```

```

        endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
        logging.info("GET http://%s", endpoint)
        response = requests.get(f"http://{endpoint}")
        logging.info("Response: %s", response.status_code)
        if response.headers.get("content-type") == "application/json":
            pp(response.json())
    elif choice == 1:
        logging.info("Checking the health of load balancer targets.")
        health =
self.loadbalancer.check_target_health(self.target_group_name)
        for target in health:
            state = target["TargetHealth"]["State"]
            logging.info(
                "Target %s on port %d is %s",
                target["Target"]["Id"],
                target["Target"]["Port"],
                state,
            )
            if state != "healthy":
                logging.warning(
                    "%s: %s",
                    target["TargetHealth"]["Reason"],
                    target["TargetHealth"]["Description"],
                )
            logging.info(
                "Note that it can take a minute or two for the health check
to update."
            )
        elif choice == 2:
            logging.info("Proceeding to the next part of the demo.")

def demo(self) -> None:
    """
    Runs the demonstration, showing how the service responds to different
failure scenarios
and how a resilient architecture can keep the service running.
    """
    ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

    logging.info("Resetting parameters to starting values for the demo.")
    self.param_helper.reset()

    logging.info(

```

```
        "Starting demonstration of the service's resilience under various
failure conditions."
    )
    self.demo_choices()

    logging.info(
        "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
    )
    self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
    logging.info("Sending GET requests will now return failure codes.")
    self.demo_choices()

    logging.info("Switching to static response mode to mitigate failure.")
    self.param_helper.put(self.param_helper.failure_response, "static")
    logging.info("Sending GET requests will now return static responses.")
    self.demo_choices()

    logging.info("Restoring normal operation of the recommendation service.")
    self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

    logging.info(
        "Introducing a failure by assigning bad credentials to one of the
instances."
    )
    self.autoscaler.create_instance_profile(
        ssm_only_policy,
        self.autoscaler.bad_creds_policy_name,
        self.autoscaler.bad_creds_role_name,
        self.autoscaler.bad_creds_profile_name,
        ["AmazonSSMManagedInstanceCore"],
    )
    instances = self.autoscaler.get_instances()
    bad_instance_id = instances[0]
    instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
    logging.info(
        "Replacing instance profile with bad credentials for instance %s.",
        bad_instance_id,
    )
    self.autoscaler.replace_instance_profile(
        bad_instance_id,
        self.autoscaler.bad_creds_profile_name,
        instance_profile["AssociationId"],
```

```

    )
    logging.info(
        "Sending GET requests may return either a valid recommendation or a
static response."
    )
    self.demo_choices()

    logging.info("Implementing deep health checks to detect unhealthy
instances.")
    self.param_helper.put(self.param_helper.health_check, "deep")
    logging.info("Checking the health of the load balancer targets.")
    self.demo_choices()

    logging.info(
        "Terminating the unhealthy instance to let the auto scaler replace
it."
    )
    self.autoscaler.terminate_instance(bad_instance_id)
    logging.info("The service remains resilient during instance
replacement.")
    self.demo_choices()

    logging.info("Simulating a complete failure of the recommendation
service.")
    self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
    logging.info(
        "All instances will report as unhealthy, but the service will still
return static responses."
    )
    self.demo_choices()
    self.param_helper.reset()

def destroy(self, automation=False) -> None:
    """
    Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
    EC2 instances, and DynamoDB table.
    """
    logging.info(
        "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
    )
    if automation:
        cleanup = True

```



```

else:
    cleanup = q.ask(
        "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
    )

if cleanup:
    logging.info("Deleting load balancer and related resources.")
    self.loadbalancer.delete_load_balancer(self.load_balancer_name)
    self.loadbalancer.delete_target_group(self.target_group_name)
    self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
    self.autoscaler.delete_key_pair()
    self.autoscaler.delete_template()
    self.autoscaler.delete_instance_profile(
        self.autoscaler.bad_creds_profile_name,
        self.autoscaler.bad_creds_role_name,
    )
    logging.info("Deleting DynamoDB table and other resources.")
    self.recommendation.destroy()
else:
    logging.warning(
        "Resources have not been deleted. Ensure you clean them up
manually to avoid unexpected charges."
    )

def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
    demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
policies and\n"

```

```
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
    for action in actions:
        if action == "deploy":
```

```

        runner.deploy()
    elif action == "demo":
        runner.demo()
    elif action == "destroy":
        runner.destroy()

logging.info("Demo completed successfully.")

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()

```

建立包裝 Auto Scaling 和 Amazon EC2 動作的類別。

```

class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.

```

```

"""
self.inst_type = inst_type
self.ami_param = ami_param
self.autoscaling_client = autoscaling_client
self.ec2_client = ec2_client
self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def create_policy(self, policy_file: str, policy_name: str) -> str:
    """
    Creates a new IAM policy or retrieves the ARN of an existing policy.

    :param policy_file: The path to a JSON file that contains the policy
    definition.
    :param policy_name: The name to give the created policy.
    :return: The ARN of the created or existing policy.
    """
    with open(policy_file) as file:
        policy_doc = file.read()

    try:
        response = self.iam_client.create_policy(
            PolicyName=policy_name, PolicyDocument=policy_doc
        )
        policy_arn = response["Policy"]["Arn"]
        log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")

```

```

        return policy_arn

    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            # If the policy already exists, get its ARN
            response = self.iam_client.get_policy(
                PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{
{policy_name}"
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
            return policy_arn
        log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
                AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(

```

```

        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
{role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

```

```

        :param policy_file: The name of a JSON file that contains the policy
        definition to
            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
        attached to
            the role, such as
        AmazonSSMManagedInstanceCore to grant
            use of Systems Manager to send commands to
        the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10) # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
            role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":

```

```

        prof_response = self.iam_client.get_instance_profile(
            InstanceProfileName=profile_name
        )
        profile_arn = prof_response["InstanceProfile"]["Arn"]
        log.info(
            "Instance profile %s already exists, nothing to do.",
profile_name
        )
        log.error(f"Full error:\n\t{err}")
        return profile_arn

def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
        if not response["IamInstanceProfileAssociations"]:
            log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")

def replace_instance_profile(
    self,
    instance_id: str,

```



```

        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
        profile is
        replaced, the instance is rebooted to ensure that it uses the new
        profile. When
        the instance is ready, Systems Manager is used to restart the Python web
        server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
        associate with
                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
        for the
                                instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )

```

```

        log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
    except ClientError as err:
        log.error("Failed to replace instance profile.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidAssociationID.NotFound":
            log.error(
                f"Association ID '{profile_association_id}' does not exist."
                "Please check the association ID and try again."
            )
        if error_code == "InvalidInstanceId":
            log.error(
                f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                f"Please verify the instance ID and try again."
            )
        log.error(f"Full error:\n\t{err}")

def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
    """
    Detaches a role from an instance profile, detaches policies from the
role,
and deletes all the resources.

:param profile_name: The name of the profile to delete.
:param role_name: The name of the role to delete.
    """
    try:
        self.iam_client.remove_role_from_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )

    self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
    log.info("Deleted instance profile %s.", profile_name)
    attached_policies = self.iam_client.list_attached_role_policies(
        RoleName=role_name
    )
    for pol in attached_policies["AttachedPolicies"]:
        self.iam_client.detach_role_policy(
            RoleName=role_name, PolicyArn=pol["PolicyArn"]
        )
        if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
            self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])

```

```

        log.info("Detached and deleted policy %s.", pol["PolicyName"])
        self.iam_client.delete_role(RoleName=role_name)
        log.info("Deleted role %s.", role_name)
    except ClientError as err:
        log.error(
            f"Couldn't delete instance profile {profile_name} or detach "
            f"policies and delete role {role_name}: {err}"
        )
        if err.response["Error"]["Code"] == "NoSuchEntity":
            log.info(
                "Instance profile %s doesn't exist, nothing to do.",
profile_name
            )

def create_key_pair(self, key_pair_name: str) -> None:
    """
    Creates a new key pair.

    :param key_pair_name: The name of the key pair to create.
    """
    try:
        response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
        with open(f"{key_pair_name}.pem", "w") as file:
            file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
        log.info("Created key pair %s.", key_pair_name)
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to create key pair {key_pair_name}.")
        if error_code == "InvalidKeyPair.Duplicate":
            log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
        log.error(f"Full error:\n\t{err}")

def delete_key_pair(self) -> None:
    """
    Deletes a key pair.
    """
    try:
        self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
        remove(f"{self.key_pair_name}.pem")
        log.info("Deleted key pair %s.", self.key_pair_name)

```

```

        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")

    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                         when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                   to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

```

```

        # Get the latest AMI ID
        ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
        ami_id = ami_latest["Parameter"]["Value"]

        # Create the launch template
        lt_response = self.ec2_client.create_launch_template(
            LaunchTemplateName=self.launch_template_name,
            LaunchTemplateData={
                "InstanceType": self.inst_type,
                "ImageId": ami_id,
                "IamInstanceProfile": {"Name": self.instance_profile_name},
                "UserData": base64.b64encode(
                    start_server_script.encode(encoding="utf-8")
                ).decode(encoding="utf-8"),
                "KeyName": self.key_pair_name,
            },
        )
        template = lt_response["LaunchTemplate"]
        log.info(
            f"Created launch template {self.launch_template_name} for AMI {ami_id} on {self.inst_type}."
        )
    except ClientError as err:
        log.error(f"Failed to create launch template {self.launch_template_name}.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
            log.info(
                f"Launch template {self.launch_template_name} already exists, nothing to do."
            )
        log.error(f"Full error:\n\t{err}")
    return template

def delete_template(self):
    """
    Deletes a launch template.
    """
    try:
        self.ec2_client.delete_launch_template(
            LaunchTemplateName=self.launch_template_name
        )

```

```

        self.delete_instance_profile(
            self.instance_profile_name, self.instance_role_name
        )
        log.info("Launch template %s deleted.", self.launch_template_name)
    except ClientError as err:
        if (
            err.response["Error"]["Code"]
            == "InvalidLaunchTemplateName.NotFoundException"
        ):
            log.info(
                "Launch template %s does not exist, nothing to do.",
                self.launch_template_name,
            )
            log.error(f"Full error:\n\t{err}")

def get_availability_zones(self) -> List[str]:
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
    except ClientError as err:
        log.error("Failed to retrieve availability zones.")
        log.error(f"Full error:\n\t{err}")
    else:
        return zones

def create_autoscaling_group(self, group_size: int) -> List[str]:
    """
    Creates an EC2 Auto Scaling group with the specified size.

    :param group_size: The number of instances to set for the minimum and
    maximum in
                        the group.
    :return: The list of Availability Zones specified for the group.
    """
    try:

```

```

        zones = self.get_availability_zones()
        self.autoscaling_client.create_auto_scaling_group(
            AutoScalingGroupName=self.group_name,
            AvailabilityZones=zones,
            LaunchTemplate={
                "LaunchTemplateName": self.launch_template_name,
                "Version": "$Default",
            },
            MinSize=group_size,
            MaxSize=group_size,
        )
        log.info(
            f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        if error_code == "AlreadyExists":
            log.info(
                f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
            )
        else:
            log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
            log.error(f"Full error:\n\t{err}")
    else:
        return zones

def get_instances(self) -> List[str]:
    """
    Gets data about the instances in the EC2 Auto Scaling group.

    :return: A list of instance IDs in the Auto Scaling group.
    """
    try:
        as_response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[self.group_name]
        )
        instance_ids = [
            i["InstanceId"]
            for i in as_response["AutoScalingGroups"][0]["Instances"]
        ]

```

```

        log.info(
            f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
        )
        if error_code == "ResourceNotFound":
            log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
        log.error(f"Full error:\n\t{err}")
    else:
        return instance_ids

def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
    """
    Terminates an instance in an EC2 Auto Scaling group. After an instance is
    terminated, it can no longer be accessed.

    :param instance_id: The ID of the instance to terminate.
    :param decrementsetting: If True, do not replace terminated instances.
    """
    try:
        self.autoscaling_client.terminate_instance_in_auto_scaling_group(
            InstanceId=instance_id,
            ShouldDecrementDesiredCapacity=decrementsetting,
        )
        log.info("Terminated instance %s.", instance_id)

        # Adding a waiter to ensure the instance is terminated
        waiter = self.ec2_client.get_waiter("instance_terminated")
        log.info("Waiting for instance %s to be terminated...", instance_id)
        waiter.wait(InstanceIds=[instance_id])
        log.info(
            f"Instance '{instance_id}' has been terminated and will be
replaced."
        )
    except ClientError as err:

```



```

        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to terminate instance '{instance_id}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to terminate the instance again."
            )
        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the resource."
            )
        log.error(f"Full error:\n\t{err}")

def attach_load_balancer_target_group(
    self, lb_target_group: Dict[str, Any]
) -> None:
    """
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    The target group specifies how the load balancer forwards requests to the
instances
    in the group.

    :param lb_target_group: Data about the ELB target group to attach.
    """
    try:
        self.autoscaling_client.attach_load_balancer_target_groups(
            AutoScalingGroupName=self.group_name,
            TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
        )
        log.info(
            "Attached load balancer target group %s to auto scaling group
%s.",
            lb_target_group["TargetGroupName"],
            self.group_name,
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."

```

```

        )
        if error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the resource."
            )
        elif error_code == "ServiceLinkedRoleFailure":
            log.error(
                "The operation failed because the service-linked role is not
ready or does not exist. "
                "Check that the service-linked role exists and is correctly
configured."
            )
        log.error(f"Full error:\n\t{err}")

def delete_autoscaling_group(self, group_name: str) -> None:
    """
    Terminates all instances in the group, then deletes the EC2 Auto Scaling
group.

:param group_name: The name of the group to delete.
    """
    try:
        response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[group_name]
        )
        groups = response.get("AutoScalingGroups", [])
        if len(groups) > 0:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, MinSize=0
            )
            instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
            for inst_id in instance_ids:
                self.terminate_instance(inst_id)

            # Wait for all instances to be terminated
            if instance_ids:
                waiter = self.ec2_client.get_waiter("instance_terminated")
                log.info("Waiting for all instances to be terminated...")
                waiter.wait(InstanceIds=instance_ids)
                log.info("All instances have been terminated.")

```

```

        else:
            log.info(f"No groups found named '{group_name}'! Nothing to do.")
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to delete the group again."
            )
        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the group."
            )
        log.error(f"Full error:\n\t{err}")

def get_default_vpc(self) -> Dict[str, Any]:
    """
    Gets the default VPC for the account.

    :return: Data about the default VPC.
    """
    try:
        response = self.ec2_client.describe_vpcs(
            Filters=[{"Name": "is-default", "Values": ["true"]}])
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error("Failed to retrieve the default VPC.")
        if error_code == "UnauthorizedOperation":
            log.error(
                "You do not have the necessary permissions to describe VPCs.
"
                "Ensure that your AWS IAM user or role has the correct
permissions."
            )
        elif error_code == "InvalidParameterValue":
            log.error(
                "One or more parameters are invalid. Check the request
parameters."

```

```

        )

        log.error(f"Full error:\n\t{err}")
    else:
        if "Vpcs" in response and response["Vpcs"]:
            log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
            return response["Vpcs"][0]
        else:
            pass

def verify_inbound_port(
    self, vpc: Dict[str, Any], port: int, ip_address: str
) -> Tuple[Dict[str, Any], bool]:
    """
    Verify the default security group of the specified VPC allows ingress
    from this
    computer. This can be done by allowing ingress from this computer's IP
    address. In some situations, such as connecting from a corporate network,
    you
    must instead specify a prefix list ID. You can also temporarily open the
    port to
    any IP address while running this example. If you do, be sure to remove
    public
    access when you're done.

    :param vpc: The VPC used by this example.
    :param port: The port to verify.
    :param ip_address: This computer's IP address.
    :return: The default security group of the specified VPC, and a value
    that indicates
        whether the specified port is open.
    """
    try:
        response = self.ec2_client.describe_security_groups(
            Filters=[
                {"Name": "group-name", "Values": ["default"]},
                {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
            ]
        )
        sec_group = response["SecurityGroups"][0]
        port_is_open = False
        log.info(f"Found default security group {sec_group['GroupId']}")

```

```

        for ip_perm in sec_group["IpPermissions"]:
            if ip_perm.get("FromPort", 0) == port:
                log.info(f"Found inbound rule: {ip_perm}")
                for ip_range in ip_perm["IpRanges"]:
                    cidr = ip_range.get("CidrIp", "")
                    if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                        port_is_open = True
                if ip_perm["PrefixListIds"]:
                    port_is_open = True
                if not port_is_open:
                    log.info(
                        f"The inbound rule does not appear to be open to
either this computer's IP "
                        f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                    )
                else:
                    break
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
        )
        if error_code == "InvalidVpcID.NotFound":
            log.error(
                f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
            )
            log.error(f"Full error:\n\t{err}")
    else:
        return sec_group, port_is_open

def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
    """
    Add an ingress rule to the specified security group that allows access on
the
specified port from the specified IP address.

:param sec_group_id: The ID of the security group to modify.
:param port: The port to open.

```

```

:param ip_address: The IP address that is granted access.
"""
try:
    self.ec2_client.authorize_security_group_ingress(
        GroupId=sec_group_id,
        CidrIp=f"{ip_address}/32",
        FromPort=port,
        ToPort=port,
        IpProtocol="tcp",
    )
    log.info(
        "Authorized ingress to %s on port %s from %s.",
        sec_group_id,
        port,
        ip_address,
    )
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
    )
    if error_code == "InvalidGroupId.Malformed":
        log.error(
            "The security group ID is malformed. "
            "Please verify that the security group ID is correct."
        )
    elif error_code == "InvalidPermission.Duplicate":
        log.error(
            "The specified rule already exists in the security group. "
            "Check the existing rules for this security group."
        )
    log.error(f"Full error:\n\t{err}")

def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.

```

```

"""
# Ensure that 'zones' is a list, even if None is passed
if zones is None:
    zones = []
try:
    paginator = self.ec2_client.get_paginator("describe_subnets")
    page_iterator = paginator.paginate(
        Filters=[
            {"Name": "vpc-id", "Values": [vpc_id]},
            {"Name": "availability-zone", "Values": zones},
            {"Name": "default-for-az", "Values": ["true"]},
        ]
    )

    subnets = []
    for page in page_iterator:
        subnets.extend(page["Subnets"])

    log.info("Found %s subnets for the specified zones.", len(subnets))
    return subnets
except ClientError as err:
    log.error(
        f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidVpcID.NotFound":
        log.error(
            "The specified VPC ID does not exist. "
            "Please check the VPC ID and try again."
        )
    # Add more error-specific handling as needed
    log.error(f"Full error:\n\t{err}")

```

建立包裝 Elastic Load Balancing 動作的類別。

```

class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

```

```

def __init__(self, elb_client: boto3.client):
    """
    Initializes the LoadBalancer class with the necessary parameters.
    """
    self.elb_client = elb_client

def create_target_group(
    self, target_group_name: str, protocol: str, port: int, vpc_id: str
) -> Dict[str, Any]:
    """
    Creates an Elastic Load Balancing target group. The target group
    specifies how
    the load balancer forwards requests to instances in the group and how
    instance
    health is checked.

    To speed up this demo, the health check is configured with shortened
    times and
    lower thresholds. In production, you might want to decrease the
    sensitivity of
    your health checks to avoid unwanted failures.

    :param target_group_name: The name of the target group to create.
    :param protocol: The protocol to use to forward requests, such as 'HTTP'.
    :param port: The port to use to forward requests, such as 80.
    :param vpc_id: The ID of the VPC in which the load balancer exists.
    :return: Data about the newly created target group.
    """
    try:
        response = self.elb_client.create_target_group(
            Name=target_group_name,
            Protocol=protocol,
            Port=port,
            HealthCheckPath="/healthcheck",
            HealthCheckIntervalSeconds=10,
            HealthCheckTimeoutSeconds=5,
            HealthyThresholdCount=2,
            UnhealthyThresholdCount=2,
            VpcId=vpc_id,
        )
        target_group = response["TargetGroups"][0]
        log.info(f"Created load balancing target group
        '{target_group_name}'.")

```



```

        return target_group
    except ClientError as err:
        log.error(
            f"Couldn't create load balancing target group
            '{target_group_name}'."
        )
        error_code = err.response["Error"]["Code"]

        if error_code == "DuplicateTargetGroupName":
            log.error(
                f"Target group name {target_group_name} already exists. "
                "Check if the target group already exists."
                "Consider using a different name or deleting the existing
target group if appropriate."
            )
        elif error_code == "TooManyTargetGroups":
            log.error(
                "Too many target groups exist in the account. "
                "Consider deleting unused target groups to create space for
new ones."
            )
            log.error(f"Full error:\n\t{err}")

def delete_target_group(self, target_group_name) -> None:
    """
    Deletes the target group.
    """
    try:
        # Describe the target group to get its ARN
        response =
self.elb_client.describe_target_groups(Names=[target_group_name])
        tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

        # Delete the target group
        self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
        log.info("Deleted load balancing target group %s.",
target_group_name)

        # Use a custom waiter to wait until the target group is no longer
available
        self.wait_for_target_group_deletion(self.elb_client, tg_arn)
        log.info("Target group %s successfully deleted.", target_group_name)

```

```

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
                log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )

```

```

def create_load_balancer(
    self,
    load_balancer_name: str,
    subnet_ids: List[str],
) -> Dict[str, Any]:
    """
    Creates an Elastic Load Balancing load balancer that uses the specified
    subnets
    and forwards requests to the specified target group.

    :param load_balancer_name: The name of the load balancer to create.
    :param subnet_ids: A list of subnets to associate with the load balancer.
    :return: Data about the newly created load balancer.
    """
    try:
        response = self.elb_client.create_load_balancer(
            Name=load_balancer_name, Subnets=subnet_ids
        )
        load_balancer = response["LoadBalancers"][0]
        log.info(f"Created load balancer '{load_balancer_name}'.")

        waiter = self.elb_client.get_waiter("load_balancer_available")
        log.info(
            f"Waiting for load balancer '{load_balancer_name}' to be
    available..."
        )
        waiter.wait(Names=[load_balancer_name])
        log.info(f"Load balancer '{load_balancer_name}' is now available!")

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to create load balancer '{load_balancer_name}'. Error
    code: {error_code}, Message: {err.response['Error']['Message']}"
        )

        if error_code == "DuplicateLoadBalancerNameException":
            log.error(
                f"A load balancer with the name '{load_balancer_name}'
    already exists. "
                "Load balancer names must be unique within the AWS region. "
                "Please choose a different name and try again."
            )

```

```

        if error_code == "TooManyLoadBalancersException":
            log.error(
                "The maximum number of load balancers has been reached in
this account and region. "
                "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
            )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer

def create_listener(
    self,
    load_balancer_name: str,
    target_group: Dict[str, Any],
) -> Dict[str, Any]:
    """
    Creates a listener for the specified load balancer that forwards requests
to the
    specified target group.

    :param load_balancer_name: The name of the load balancer to create a
listener for.
    :param target_group: An existing target group that is added as a listener
to the
                           load balancer.
    :return: Data about the newly created listener.
    """
    try:
        # Retrieve the load balancer ARN
        load_balancer_response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        load_balancer_arn = load_balancer_response["LoadBalancers"][0][
            "LoadBalancerArn"
        ]

        # Create the listener
        response = self.elb_client.create_listener(
            LoadBalancerArn=load_balancer_arn,
            Protocol=target_group["Protocol"],
            Port=target_group["Port"],
            DefaultActions=[

```

```

        {
            "Type": "forward",
            "TargetGroupArn": target_group["TargetGroupArn"],
        }
    ],
)
log.info(
    f"Created listener to forward traffic from load balancer
    '{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
)
return response["Listeners"][0]
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to add a listener on '{load_balancer_name}' for target
        group '{target_group['TargetGroupName']}'."
    )

    if error_code == "ListenerNotFoundException":
        log.error(
            f"The listener could not be found for the load balancer
            '{load_balancer_name}'. "
            "Please check the load balancer name and target group
            configuration."
        )
    if error_code == "InvalidConfigurationRequestException":
        log.error(
            f"The configuration provided for the listener on load
            balancer '{load_balancer_name}' is invalid. "
            "Please review the provided protocol, port, and target group
            settings."
        )
    log.error(f"Full error:\n\t{err}")

def delete_load_balancer(self, load_balancer_name) -> None:
    """
    Deletes a load balancer.

    :param load_balancer_name: The name of the load balancer to delete.
    """
    try:
        response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]

```

```

    )
    lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
    self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
    log.info("Deleted load balancer %s.", load_balancer_name)
    waiter = self.elb_client.get_waiter("load_balancers_deleted")
    log.info("Waiting for load balancer to be deleted...")
    waiter.wait(Names=[load_balancer_name])
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
    )

    if error_code == "LoadBalancerNotFoundException":
        log.error(
            f"The load balancer '{load_balancer_name}' does not exist. "
            "Please check the name and try again."
        )
    log.error(f"Full error:\n\t{err}")

def get_endpoint(self, load_balancer_name) -> str:
    """
    Gets the HTTP endpoint of the load balancer.

    :return: The endpoint.
    """
    try:
        response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        return response["LoadBalancers"][0]["DNSName"]
    except ClientError as err:
        log.error(
            f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "LoadBalancerNotFoundException":
            log.error(
                "Verify load balancer name and ensure it exists in the AWS
console."
            )

```

```

        log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
        balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )

```

```

        health_response = self.elb_client.describe_target_health(
            TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
        )
    except ClientError as err:
        log.error(f"Couldn't check health of {target_group_name} target(s).")
        error_code = err.response["Error"]["Code"]
        if error_code == "LoadBalancerNotFoundException":
            log.error(
                "Load balancer associated with the target group was not
found. "
                "Ensure the load balancer exists, is in the correct AWS
region, and "
                "that you have the necessary permissions to access it.",
            )
        elif error_code == "TargetGroupNotFoundException":
            log.error(
                "Target group was not found. "
                "Verify the target group name, check that it exists in the
correct region, "
                "and ensure it has not been deleted or created in a different
account.",
            )
        log.error(f"Full error:\n\t{err}")
    else:
        return health_response["TargetHealthDescriptions"]

```

建立使用 DynamoDB 模擬建議服務的類別。

```

class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
    movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
        parameters.

```



```

:param table_name: The name of the DynamoDB recommendations table.
:param dynamodb_client: A Boto3 DynamoDB client.
"""

self.table_name = table_name
self.dynamodb_client = dynamodb_client

def create(self) -> Dict[str, Any]:
    """
    Creates a DynamoDB table to use as a recommendation service. The table
    has a
    hash key named 'MediaType' that defines the type of media recommended,
    such as
    Book or Movie, and a range key named 'ItemId' that, combined with the
    MediaType,
    forms a unique identifier for the recommended item.

    :return: Data about the newly created table.
    :raises RecommendationServiceError: If the table creation fails.
    """
    try:
        response = self.dynamodb_client.create_table(
            TableName=self.table_name,
            AttributeDefinitions=[
                {"AttributeName": "MediaType", "AttributeType": "S"},
                {"AttributeName": "ItemId", "AttributeType": "N"},
            ],
            KeySchema=[
                {"AttributeName": "MediaType", "KeyType": "HASH"},
                {"AttributeName": "ItemId", "KeyType": "RANGE"},
            ],
            ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
        )
        log.info("Creating table %s...", self.table_name)
        waiter = self.dynamodb_client.get_waiter("table_exists")
        waiter.wait(TableName=self.table_name)
        log.info("Table %s created.", self.table_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceInUseException":
            log.info("Table %s exists, nothing to be done.", self.table_name)
        else:
            raise RecommendationServiceError(
                self.table_name, f"ClientError when creating table: {err}."
            )

```

```

        )
    else:
        return response

def populate(self, data_file: str) -> None:
    """
    Populates the recommendations table from a JSON file.

    :param data_file: The path to the data file.
    :raises RecommendationServiceError: If the table population fails.
    """
    try:
        with open(data_file) as data:
            items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch}))
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}")
    )

def destroy(self) -> None:
    """
    Deletes the recommendations table.

    :raises RecommendationServiceError: If the table deletion fails.
    """
    try:
        self.dynamodb_client.delete_table(TableName=self.table_name)
        log.info("Deleting table %s...", self.table_name)
        waiter = self.dynamodb_client.get_waiter("table_not_exists")
        waiter.wait(TableName=self.table_name)
        log.info("Table %s deleted.", self.table_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            log.info("Table %s does not exist, nothing to do.",
self.table_name)
        else:

```

```

        raise RecommendationServiceError(
            self.table_name, f"ClientError when deleting table: {err}."
        )

```

建立包裝 Systems Manager 動作的類別。

```

class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
    to drive
    the demonstration of resilient architecture, such as failure of a dependency
    or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
        recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
        a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")

```

```

        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")

```

- 如需 API 詳細資訊，請參閱《適用於 Python (Boto3) 的 AWS SDK API 參考》中的下列主題。
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)

- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

對 Amazon EC2 Auto Scaling 中的問題進行故障診斷

Amazon EC2 Auto Scaling 提供特定和描述性錯誤，協助您進行故障診斷。您可以在描述擴展活動中找到錯誤訊息。

主題

- [從擴展活動擷取錯誤訊息](#)
- [關閉擴展活動](#)
- [其他疑難排解資源](#)
- [Amazon EC2 Auto Scaling 故障診斷：EC2 執行個體啟動失敗](#)
- [Amazon EC2 Auto Scaling 故障診斷：AMI 問題](#)
- [Amazon EC2 Auto Scaling 故障診斷：負載平衡器問題](#)
- [Amazon EC2 Auto Scaling 的故障診斷：啟動範本](#)

從擴展活動擷取錯誤訊息

若要從擴展活動的描述中擷取錯誤訊息，請使用 [describe-scaling-activities](#) 命令。您有可追溯到 6 週前的擴展活動紀錄。擴展活動依開始時間排序，並會先列出最新的擴展活動。

Note

擴展活動也會顯示在 Amazon EC2 Auto Scaling 主控台的 Auto Scaling 群組的 Activity (活動) 索引標籤的活動歷史記錄中。

若要查看特定 Auto Scaling 群組的擴展活動，請使用下列命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下是範例回應，其中 StatusCode 包含目前活動的狀態，StatusMessage 包含錯誤訊息。

```
{
  "Activities": [
    {
```

```

        "ActivityId": "3b05dbf6-037c-b92f-133f-38275269dc0f",
        "AutoScalingGroupName": "my-asg",
        "Description": "Launching a new EC2 instance: i-003a5b3ffe1e9358e. Status
Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with
token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle Action Completed
with ABANDON Result",
        "Cause": "At 2021-01-11T00:35:52Z a user request created an
AutoScalingGroup changing the desired capacity from 0 to 1. At 2021-01-11T00:35:53Z
an instance was started in response to a difference between desired and actual
capacity, increasing the capacity from 0 to 1.",
        "StartTime": "2021-01-11T00:35:55.542Z",
        "EndTime": "2021-01-11T01:06:31Z",
        "StatusCode": "Cancelled",
        "StatusMessage": "Instance failed to complete user's Lifecycle Action:
Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned:
Lifecycle Action Completed with ABANDON Result",
        "Progress": 100,
        "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}\",
        "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
    ...
]
}

```

如需輸出中欄位的說明，請參閱《Amazon EC2 Auto Scaling API 參考》中的[活動](#)。

檢視已刪除群組的擴展活動

若要在刪除 Auto Scaling 群組之後檢視擴展活動，請按如下所示將 `--include-deleted-groups` 選項新增至 [describe-scaling-activities](#) 命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg --include-deleted-groups
```

以下為回應範例，其中包含已刪除群組的擴展活動。

```

{
  "Activities": [
    {

```

```

        "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
        "AutoScalingGroupName": "my-asg",
        "Description": "Launching a new EC2 instance. Status Reason: Your Spot
request price of 0.001 is lower than the minimum required Spot request fulfillment
price of 0.0031. Launching EC2 instance failed.",
        "Cause": "At 2021-01-13T20:47:24Z a user request update of AutoScalingGroup
constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 0 to 3.
At 2021-01-13T20:47:27Z an instance was started in response to a difference between
desired and actual capacity, increasing the capacity from 0 to 3.",
        "StartTime": "2021-01-13T20:47:30.094Z",
        "EndTime": "2021-01-13T20:47:30Z",
        "StatusCode": "Failed",
        "StatusMessage": "Your Spot request price of 0.001 is lower than the
minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance
failed.",
        "Progress": 100,
        "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}\",
        "AutoScalingGroupState": "Deleted",
        "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
    ...
]
}

```

關閉擴展活動

如果您需要在不干擾擴展政策或排程動作的情況下調查問題，您可以使用下列選項：

- 透過暫停和 AlarmNotification ScheduledActions 程序，防止所有動態擴展政策和排程動作變更群組所需的容量。如需詳細資訊，請參閱[暫停和繼續 Amazon EC2 Auto Scaling 程序](#)。
- 停用個別動態擴展政策，使其不會變更群組的所需容量，以回應負載變更。如需詳細資訊，請參閱[停用 Auto Scaling 群組的擴展政策](#)。
- 透過停用政策的縮減部分，將個別目標追蹤擴展政策更新為僅橫向擴展（新增容量）。此方法可防止群組所需的容量縮減，但允許在負載增加時增加。如需詳細資訊，請參閱[Amazon EC2 Auto Scaling 的目標追蹤擴展政策](#)。
- 更新您的預測擴展政策以僅預測模式。在僅限預測模式下，預測擴展將繼續產生預測，但不會自動增加容量。如需詳細資訊，請參閱[建立 Auto Scaling 群組的預測擴展政策](#)。

其他疑難排解資源

以下頁面提供針對 Amazon EC2 Auto Scaling 問題進行疑難排解的其他資訊。

- [驗證 Auto Scaling 群組的擴展活動](#)
- [在 Amazon EC2 Auto Scaling 主控台中檢視監控圖表](#)
- [Auto Scaling 群組中的執行個體運作狀態檢查](#)
- [lifecycle hook 的考量與限制](#)
- [在 Auto Scaling 群組中完成生命週期動作](#)
- [使用 Amazon VPC 為您的 Auto Scaling 執行個體提供網際網路連線](#)
- [從 Auto Scaling 群組暫時移除執行個體](#)
- [停用 Auto Scaling 群組的擴展政策](#)
- [暫停和繼續 Amazon EC2 Auto Scaling 程序](#)
- [控制縮減期間應終止的 Auto Scaling 執行個體](#)
- [刪除 Auto Scaling 基礎設施](#)
- [Auto Scaling 資源和群組的配額](#)

下列 AWS 資源也可以提供幫助：

- [AWS 知識中心的 Amazon EC2 Auto Scaling 主題](#)
- [AWS re : Post 上的 Amazon EC2 Auto Scaling 問題](#)
- [AWS 運算部落格中的 Amazon EC2 Auto Scaling 文章](#)
- [《CloudFormation 使用者指南》中的 CloudFormation](#)

疑難排解通常需要專家或小幫手社群的反覆查詢和探索。如果您在嘗試本節中的建議後仍遇到問題，請聯絡 AWS 支援（在 AWS 管理主控台，按一下支援、支援中心）或使用 Amazon EC2 Auto Scaling 標籤在 [AWS re : Post](#) 上提出問題。

Amazon EC2 Auto Scaling 故障診斷：EC2 執行個體啟動失敗

此頁面提供有關您的 EC2 執行個體啟動失敗、可能的原因，以及您可以採取用以解決問題之步驟的資訊。

若要擷取錯誤訊息，請參閱 [從擴展活動擷取錯誤訊息](#)。

當您的 EC2 執行個體無法啟動時，您可能取得以下一或多個錯誤訊息：

啟動問題

- [目前並不支援要求的組態。](#)
- [安全群組 \(安全群組的名稱\) 不存在。啟動 EC2 執行個體失敗。](#)
- [金鑰對 \(與您的 EC2 執行個體關聯的金鑰對\) 不存在。啟動 EC2 執行個體失敗。](#)
- [您請求的執行個體類型 \(<執行個體類型>\) 在您請求的可用區域 \(執行個體可用區域\) 並不受支援...](#)
- [您的 Spot 請求價格 0.015 低於 Spot 請求所需的最低履行價格 0.0735...](#)
- [裝置名稱 <device name> 無效/裝置名稱上傳無效。啟動 EC2 執行個體失敗。](#)
- [適用於參數 virtualName 的值 \(與執行個體儲存體裝置關聯的名稱\) 無效... 啟動 EC2 執行個體失敗。](#)
- [執行個體存放區 AMI 不支援 EBS 區塊型設備映射。](#)
- [置放群組可能無法與「<執行個體類型>」搭配使用。啟動 EC2 執行個體失敗。](#)
- [Client.InternalError：啟動時的用戶端錯誤。](#)
- [我們目前在您要求的可用區域中，並沒有足夠的 \(執行個體類型\) 容量... 啟動 EC2 執行個體失敗。](#)
- [請求的保留沒有足夠的相容性和可用容量來滿足此要求。啟動 EC2 執行個體失敗。](#)
- [您的容量區塊保留 <保留 id> 尚未啟動。啟動 EC2 執行個體失敗。](#)
- [沒有符合請求的 Spot 容量。啟動 EC2 執行個體失敗。](#)
- [\(執行個體數\) 個執行個體已在運作中。啟動 EC2 執行個體失敗。](#)

目前並不支援要求的組態。

原因：啟動範本或啟動組態中的某些選項可能與執行個體類型不相容，或您請求 AWS 的區域或可用區域可能不支援執行個體組態。

解決方案：嘗試不同的執行個體組態。若要搜尋符合您需求的執行個體類型，請參閱《[Amazon EC2 使用者指南](#)》中的[尋找 Amazon EC2 執行個體類型](#)。Amazon EC2

如需解決此問題的進一步指導，請檢查下列項目：

- 確保您已選擇執行個體類型支援的 AMI。例如，如果執行個體類型使用 Arm 型 AWS Graviton 處理器而非 Intel Xeon 處理器，則需要 Arm 相容的 AMI。如需選擇相容執行個體類型的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[變更執行個體類型的相容性](#)。
- 測試所請求區域和可用區域中是否有可用的執行個體類型。最新一代的執行個體類型也許尚無法用於特定的區域或可用區域。較舊一代的執行個體類型也許尚無法用於較新的區域或可用區域。若要依

- 置 (區域或可用區域) 搜尋提供的執行個體類型，請使用 [describe-instance-type-offerings](#)。如需詳細資訊，請參閱《[Amazon EC2 使用者指南](#)》中的尋找 [Amazon EC2 執行個體類型](#)。Amazon EC2
- 如果使用的是專用執行個體或專用主機，務必選擇作為專用執行個體或專用主機支援的執行個體類型。

安全群組 (安全群組的名稱) 不存在。啟動 EC2 執行個體失敗。

原因：啟動範本或啟動組態中指定的安全群組可能已遭刪除。

解決方案：

1. 使用 [describe-security-groups](#) 命令，取得與您的帳戶相關聯的安全群組清單。
2. 從清單中選取要使用的安全群組。若要改為建立安全群組，請使用 [create-security-group](#) 命令。
3. 建立新的啟動範本或啟動組態。
4. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

金鑰對 (與您的 EC2 執行個體關聯的金鑰對) 不存在。啟動 EC2 執行個體失敗。

原因：啟動執行個體時所使用的金鑰對可能已遭刪除。

解決方案：

1. 使用 [describe-key-pairs](#) 命令以取得您可用的金鑰對清單。
2. 從清單中選取要使用的金鑰對。若要改為建立金鑰對，請使用 [create-key-pair](#) 命令。
3. 建立新的啟動範本或啟動組態。
4. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

您請求的執行個體類型 (<執行個體類型>) 在您請求的可用區域 (執行個體可用區域) 並不受支援...

錯誤訊息：您請求的執行個體類型 (<執行個體類型>) 在您請求的可用區域 (<執行個體可用區域>) 並不受支援...啟動 EC2 執行個體失敗。

原因：Auto Scaling 群組所指定的可用區域不支援您選擇的執行個體類型。

解決方案：

1. 使用描述 [describe-instance-type-offerings](#) 命令或 Amazon EC2 主控台，在執行個體類型頁面的聯網窗格上檢查可用區域值，以確認哪些可用區域支援您選擇的執行個體類型。
2. 使用 [update-auto-scaling-group](#) 命令，更新或移除 Auto Scaling 群組的設定中任何不支援區域的子網路。如需詳細資訊，請參閱[新增可用區域](#)。

您的 Spot 請求價格 0.015 低於 Spot 請求所需的最低履行價格 0.0735...

原因：您請求的 Spot 最高價格低於您所選執行個體類型的 Spot 價格。

解決方案：以較高的 Spot 最高價格 (可能是隨需價格) 來提交新請求。之前，您支付的 Spot 價格是以出價為基礎。現在，您支付的是現行 Spot 價格。將最高價格設得更高，可讓 Amazon EC2 Spot 服務更好地啟動和維護所需容量。

裝置名稱 <device name> 無效/裝置名稱上傳無效。啟動 EC2 執行個體失敗。

原因 1：您的啟動範本或啟動組態中的區塊型設備映射，可能包含無法使用或目前未受支援的區塊型儲存設備名稱。

解決方案：

1. 驗證哪些裝置名稱可用於您的特定執行個體組態。如需裝置命名的詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [Linux 執行個體上的裝置名稱](#)。
2. 手動建立不屬於 Auto Scaling 群組的 Amazon EC2 執行個體，並調查問題。如果區塊型儲存設備命名組態與 Amazon Machine Image (AMI) 中的名稱衝突，則執行個體會啟動失敗。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[封鎖裝置映射](#)。
3. 確認您已成功啟動執行個體後，請使用 [describe-volumes](#) 命令，以了解磁碟區提供給執行個體的資訊。
4. 使用磁碟區描述所列的裝置名稱來建立新的啟動範本或啟動組態。
5. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

適用於參數 `virtualName` 的值 (與執行個體儲存體裝置關聯的名稱) 無效... 啟動 EC2 執行個體失敗。

原因：為與區塊型儲存設備關聯的虛擬名稱指定的格式不正確。

解決方案：

1. 透過以 `virtualName` 參數指定裝置名稱來建立新的啟動範本或啟動組態。如需有關裝置名稱格式的資訊，請參閱《Amazon EC2 使用者指南》中的 [Linux 執行個體上的裝置命名](#)。
2. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

執行個體存放區 AMI 不支援 EBS 區塊型設備映射。

原因：您的執行個體上不支援啟動範本或啟動組態中所指定的區塊型設備映射。

解決方案：

1. 使用執行個體類型支援的區塊型設備映射來建立新的啟動範本或啟動組態。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [封鎖裝置映射](#)。
2. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

置放群組可能無法與「<執行個體類型>」搭配使用。啟動 EC2 執行個體失敗。

原因：您的叢集置放群組包含無效的執行個體類型。

解決方案：

1. 如需置放群組支援之有效執行個體類型的相關資訊，請參閱《Amazon EC2 使用者指南》中的 [置放群組](#)。
2. 遵照 [置放群組](#) 中的詳細指示來建立新的置放群組。
3. 或者，使用受支援的執行個體類型建立新的啟動範本或啟動組態。
4. 使用 [update-auto-scaling-group](#) 命令，利用新的置放群組、啟動範本或啟動組態來更新您的 Auto Scaling 群組。

Client.InternalError：啟動時的用戶端錯誤。

問題：Amazon EC2 Auto Scaling 會嘗試啟動具有加密 EBS 磁碟區的執行個體，但服務連結角色無法存取用來加密它 AWS KMS 的客戶受管金鑰。如需詳細資訊，請參閱[與加密磁碟區搭配使用的必要 AWS KMS 金鑰政策](#)。

原因 1：您需要能授予許可的金鑰政策，才能將客戶受管金鑰用於適當的服務連結角色。

解決方案 1：允許服務連結角色使用客戶受管金鑰，如下所示：

1. 決定將哪個服務連結角色用於這個 Auto Scaling 群組。
2. 更新客戶受管金鑰的金鑰政策，並允許服務連結角色使用客戶受管金鑰。
3. 更新 Auto Scaling 群組以使用服務連結角色。

如需可讓服務連結角色使用客戶受管金鑰的金鑰政策範例，請參閱[範例 1：允許存取客戶受管金鑰的金鑰政策區段](#)。

原因 2：如果客戶受管金鑰和 Auto Scaling 群組位於不同的 AWS 帳戶中，您需要設定客戶受管金鑰的跨帳戶存取權，以便授予許可，以使用客戶受管金鑰來取得適當的服務連結角色。

解決方案 2：允許外部帳戶中的服務連結角色在本機帳戶中使用客戶受管金鑰，如下所示：

1. 更新針對客戶受管金鑰的金鑰政策，以便允許 Auto Scaling 群組帳戶存取客戶受管金鑰。
2. 在可以建立授權的 Auto Scaling 群組帳戶內定義 IAM 使用者或角色。
3. 決定將哪個服務連結角色用於這個 Auto Scaling 群組。
4. 利用適當的服務連結角色以被授予者委託人身分建立對客戶受管金鑰的授權。
5. 更新 Auto Scaling 群組以使用服務連結角色。

如需詳細資訊，請參閱[範例 2：允許跨帳戶存取客戶受管金鑰的金鑰政策區段](#)。

解決方案 3：在和 Auto Scaling 群組相同的 AWS 帳戶中使用客戶受管金鑰。

1. 使用屬於和 Auto Scaling 群組相同的帳戶的另一個客戶受管金鑰來複製和重新加密快照。
2. 允許服務連結角色使用新的客戶受管金鑰。請參閱解決方案 1 的步驟。

我們目前在您要求的可用區域中，並沒有足夠的 (執行個體類型) 容量... 啟動 EC2 執行個體失敗。

錯誤訊息：在您請求的可用區域 (請求可用區域) 中，目前並沒有足夠的 (執行個體類型) 容量。我們的系統將會處理佈建額外的容量。您目前可以取得 (執行個體類型) 容量，做法是不在您的要求中指定可用區域，或選擇 (目前支援執行個體類型的可用區域清單)。啟動 EC2 執行個體失敗。

原因：目前不支援請求的執行個體類型和可用區域組合。

解決方案：若要解決問題，請嘗試下列動作：

- 請等待幾分鐘，讓 Amazon EC2 Auto Scaling 在其他已啟用的可用區域中尋找此執行個體類型的容量。
- 將 Auto Scaling 群組擴展到其他可用區域。如需詳細資訊，請參閱[新增可用區域](#)。
- 請遵循使用多種執行個體類型的最佳實務，以免您依賴某個特定執行個體類型。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

請求的保留沒有足夠的相容性和可用容量來滿足此要求。啟動 EC2 執行個體失敗。

原因 1：您已達到可使用 targeted 隨需容量保留啟動執行個體的數目上限。

解決方案 1：增加您可以透過 targeted 隨需容量保留啟動的執行個體數量，或使用容量保留群組，讓預留容量以一般隨需容量的形式啟動。如需詳細資訊，請參閱[使用容量預留在特定可用區域中預留容量](#)。

原因 2：您已達到可使用容量區塊啟動執行個體的數目上限。

使用容量區塊時，您會受到原始購買容量的限制。如果您遇到啟動次數高於預期，且用盡了所有可用的容量，則會造成啟動失敗。執行個體在完全終止之前會經過冗長的清理過程。在此期間它們不能被重複使用。這也可能造成啟動失敗。如需詳細資訊，請參閱[Capacity Blocks 用於機器學習工作負載](#)。

解決方案 2：若要解決問題，請嘗試下列動作：

- 請保持請求不變。如果容量區塊執行個體正在終止，您必須等待幾分鐘，執行個體才能完成終止，且容量會再次變成可用。Amazon EC2 Auto Scaling 會繼續自動發出啟動請求，直到有可用的容量出現。
- 請確保您購買了足夠的容量來應付尖峰工作負載，以免頻繁遇到此錯誤。

您的容量區塊保留 <保留 id> 尚未啟動。啟動 EC2 執行個體失敗。

原因：指定的容量區塊尚未啟動。

解決方案：遵循建議的容量區塊方法，並使用排程擴展。這樣做有助於確保僅在啟動保留時才增加 Auto Scaling 群組的所需容量，並在保留結束之前減少所需容量。

沒有符合請求的 Spot 容量。啟動 EC2 執行個體失敗。

原因：目前沒有足夠的備用容量來滿足您對 Spot 執行個體的請求。

解決方案：若要解決問題，請嘗試下列動作：

- 等候幾分鐘，因為容量會頻繁變化。Amazon EC2 Auto Scaling 會繼續自動發出啟動請求，直到有可用的容量出現。
- 將 Auto Scaling 群組擴展到其他可用區域。如需詳細資訊，請參閱[新增可用區域](#)。
- 請遵循使用多種執行個體類型的最佳實務，以免您依賴某個特定執行個體類型。如需詳細資訊，請參閱[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)。

(執行個體數) 個執行個體已在運作中。啟動 EC2 執行個體失敗。

原因：您已達到可在一個區域中啟動執行個體的數目上限。當您建立 AWS 帳戶時，我們會針對每個區域可執行的執行個體數量設定預設限制。

解決方案：若要解決問題，請嘗試下列動作：

- 如果目前限制無法滿足需求，您可以按區域請求增加配額。如需詳細資訊，請參閱[《Amazon EC2 使用者指南》中的 Amazon EC2 服務配額](#)。Amazon EC2
- 提交減少了執行個體數量的新請求 (可在後續階段增加數量)。

Amazon EC2 Auto Scaling 故障診斷：AMI 問題

此頁面提供有關您的 AMI 相關問題、可能的原因，以及您可以採取用以解決問題之步驟的資訊。

若要擷取錯誤訊息，請參閱[從擴展活動擷取錯誤訊息](#)。

當您的 EC2 執行個體因為 AMI 的問題而無法啟動時，您可能會收到以下一或多個錯誤訊息。

AMI 問題

- [AMI ID \(您的 AMI 的 ID\) 不存在。啟動 EC2 執行個體失敗。](#)
- [AMI \(AMI ID\) 等待處理中，無法執行。啟動 EC2 執行個體失敗。](#)
- [無效的裝置名稱 <裝置名稱>。啟動 EC2 執行個體失敗。](#)
- [指定執行個體類型的架構「arm64」與指定 AMI 的架構「x86_64」不相符...啟動 EC2 執行個體失敗。](#)
- [AMI「<AMI ID>」已停用，無法執行。啟動 EC2 執行個體失敗。](#)

Important

AWS 支援透過修改 AMI 許可，私下與其他 AWS 帳戶共用 AMI。如果 AMI 在未共享的情況下設為私有，則在啟動新的執行個體時可能會導致授權錯誤。如需共用私有 AMIs 的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[與特定 AWS 帳戶共用 AMI](#)。

AMI ID (您的 AMI 的 ID) 不存在。啟動 EC2 執行個體失敗。

- 原因：AMI 可能已在建立啟動範本或啟動組態後刪除。
- 解決方案：
 1. 使用有效的 AMI 建立新的啟動範本或啟動組態。
 2. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

AMI (AMI ID) 等待處理中，無法執行。啟動 EC2 執行個體失敗。

原因：您可能剛建立 AMI (透過取得執行中執行個體的快照或其他任何方法)，並且其可能尚無法使用。

解決方案：您必須等到 AMI 可用，才能建立啟動範本或啟動組態。

無效的裝置名稱 <裝置名稱>。啟動 EC2 執行個體失敗。

原因：將 EBS 磁碟區連接到 EC2 執行個體時，必須為該磁碟區提供有效的裝置名稱。所選取的 AMI 必須支援此裝置名稱。

解決方案：

1. 建立新的啟動範本或啟動組態，並未 AMI 指定正確的裝置名稱。建議的命名慣例會根據 AMI 的虛擬化類型而異。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[裝置名稱](#)。

2. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

指定執行個體類型的架構「arm64」與指定 AMI 的架構「x86_64」不相符... 啟動 EC2 執行個體失敗。

原因 1：如果 AMI 的架構和啟動範本或啟動組態中使用的執行個體類型不相同，則當 Amazon EC2 Auto Scaling 嘗試使用不相容的執行個體組態啟動執行個體時會收到錯誤訊息。

解決方案 1：

1. 使用 [describe-images](#) 命令或 Amazon EC2 主控台，在 Amazon Machine Images (AMI) 頁面的詳細資料窗格中檢查架構值，以驗證您的 AMI。
2. 使用 [describe-instance-types](#) 命令或 Amazon EC2 主控台，檢查執行個體類型畫面上的架構欄，以尋找與 AMI 架構相同的執行個體類型。如需選擇相容執行個體類型的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[變更執行個體類型的相容性](#)。
3. 使用與您的 AMI 具有相同架構的執行個體類型來建立新的啟動範本或啟動組態。
4. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

原因 2：Amazon EC2 Auto Scaling 會嘗試為 Auto Scaling 群組啟動混合執行個體政策中指定的執行個體類型，但執行個體類型與啟動範本中指定的 AMI 具有不同的架構。

解決方案 1：請勿在混合執行個體政策中包含具有不同架構的執行個體類型。

1. 使用 [describe-images](#) 命令或 Amazon EC2 主控台，在 Amazon Machine Images (AMI) 頁面的詳細資料窗格中檢查架構值，以驗證您的 AMI。
2. 使用 [describe-instance-types](#) 命令或 Amazon EC2 主控台，檢查執行個體類型畫面上的架構欄，驗證要包含在混合執行個體政策中的每個執行個體類型的架構。如需選擇相容執行個體類型的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[變更執行個體類型的相容性](#)。
3. 使用 [update-auto-scaling-group](#) 命令，從 Auto Scaling 群組中更新或移除不相容的執行個體類型。

解決方案 2：若要在相同的 Auto Scaling 群組中啟動 Arm (Graviton2) 和 x86_64 (Intel) 執行個體，您必須分別使用 ARM 相容 AMI 和 Intel x86 相容 AMI 支援的啟動範本，以符合混合執行個體政策中的執行個體類型。

1. 使用 [describe-images](#) 命令或 Amazon EC2 主控台，在 Amazon Machine Images (AMI) 頁面的詳細資料窗格中檢查架構值，以驗證現有啟動範本中 AMI 的架構。
2. 使用與您打算使用的其他架構相符的 AMI 建立新的啟動範本。
3. 使用 [update-auto-scaling-group](#) 命令更新您的 Auto Scaling 群組，以覆寫現有的啟動範本，並為每個相容的執行個體類型指定新的啟動範本。如需詳細資訊，請參閱[使用多個啟動範本](#)。

AMI 「<AMI ID>」已停用，無法執行。啟動 EC2 執行個體失敗。

原因：您試圖從已停用的 AMI 啟動執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[停用 AMI](#)。

解決方案：

1. 建立新的啟動範本或啟動組態，並指定未停用的 AMI。
2. 使用 [update-auto-scaling-group](#) 命令，利用新的啟動範本或啟動組態來更新您的 Auto Scaling 群組。

Amazon EC2 Auto Scaling 故障診斷：負載平衡器問題

此頁面提供與您 Auto Scaling 群組關聯之負載平衡器所造成的問題、可能原因，以及您可以採取用以解決問題之步驟的相關資訊。

若要擷取錯誤訊息，請參閱 [從擴展活動擷取錯誤訊息](#)。

當 EC2 執行個體因與 Auto Scaling 群組關聯的負載平衡器問題而無法啟動時，您可能會收到以下一或多條錯誤訊息。

負載平衡器問題

- [未找到一個或多個目標群組。驗證負載平衡器組態失敗。](#)
- [找不到負載平衡器 <您的負載平衡器>。驗證負載平衡器組態失敗。](#)
- [沒有名為 \(負載平衡器名稱\) 的作用中負載平衡器。更新負載平衡器組態失敗。](#)
- [EC2 執行個體 \(執行個體 ID\) 不在 VPC 中。更新負載平衡器組態失敗。](#)

Note

檢查 Auto Scaling 群組中的執行個體是否可透過負載平衡器進行存取，使用 Reachability Analyzer 來疑難排解連線問題。若要了解 Reachability Analyzer 自動偵測到的不同網路設定錯誤問題，請參閱《Reachability Analyzer 使用者指南》中的 [Reachability Analyzer 說明程式碼](#)。

未找到一個或多個目標群組。驗證負載平衡器組態失敗。

問題：當您的 Auto Scaling 群組啟動執行個體時，Amazon EC2 Auto Scaling 會嘗試驗證與 Auto Scaling 群組關聯的 Elastic Load Balancing 資源是否存在。當找不到目標群組時，擴展活動會失敗，並且您會看到 One or more target groups not found. Validating load balancer configuration failed. 錯誤。

原因 1：已刪除連接至 Auto Scaling 群組的目標群組。

解決方案 1：您可以使用 Amazon EC2 Auto Scaling 主控台或 [detach-load-balancer-target-groups](#) 命令，建立沒有目標群組的新 Auto Scaling 群組，或者從 Auto Scaling 群組中移除未使用的目標群組。

原因 2：目標群組存在，但是在建立 Auto Scaling 群組時，嘗試指定目標群組 ARN 會發生問題。未按正確順序建立資源。

解決方案 2：建立新的 Auto Scaling 群組，並在最後指定目標群組。

找不到負載平衡器 <您的負載平衡器>。驗證負載平衡器組態失敗。

問題：當您的 Auto Scaling 群組啟動執行個體時，Amazon EC2 Auto Scaling 會嘗試驗證與 Auto Scaling 群組關聯的 Elastic Load Balancing 資源是否存在。當找不到 Classic Load Balancer 時，擴展活動會失敗，並且您會得到 Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed. 錯誤。

原因 1：已刪除 Classic Load Balancer。

解決方案 1：您可以使用 Amazon EC2 Auto Scaling 主控台或 [detach-load-balancers](#) 命令，建立沒有負載平衡器的新 Auto Scaling 群組，或者從 Auto Scaling 群組中移除未使用的負載平衡器。

原因 2：Classic Load Balancer 存在，但當建立 Auto Scaling 群組時，嘗試指定負載平衡器名稱會發生問題。未按正確順序建立資源。

解決方案 2：建立新的 Auto Scaling 群組，並在最後面指定負載平衡器名稱。

沒有名為 (負載平衡器名稱) 的作用中負載平衡器。更新負載平衡器組態失敗。

原因：指定的負載平衡器可能已刪除。

解決方案：您可以建立新的負載平衡器，然後建立新的 Auto Scaling 群組，或者建立不帶負載平衡器的新 Auto Scaling 群組。

EC2 執行個體 (執行個體 ID) 不在 VPC 中。更新負載平衡器組態失敗。

原因：指定的執行個體不存在於 VPC。

解決方案：您可以刪除與執行個體關聯的負載平衡器，或建立新 Auto Scaling 群組。

Amazon EC2 Auto Scaling 的故障診斷：啟動範本

請使用以下資訊來協助您診斷和修復在嘗試指定 Auto Scaling 群組的啟動範本時發生的常見問題。

無法啟動執行個體

如果您無法使用已指定的啟動範本啟動任何執行個體，請檢查下列項目以進行一般故障診斷：[Amazon EC2 Auto Scaling 故障診斷：EC2 執行個體啟動失敗](#)。

您必須使用有效的完整格式啟動範本 (無效值)

問題：在嘗試為 Auto Scaling 群組指定啟動範本時，您收到 You must use a valid fully-formed launch template 錯誤。您可能會遇到此錯誤，因為只有在建立或更新使用啟動範本的 Auto Scaling 群組時才會驗證啟動範本中的值。

原因 1：若您收到 You must use a valid fully-formed launch template 錯誤，則存在導致 Amazon EC2 Auto Scaling 考慮有關啟動範本無效的問題。這是一個通用錯誤，可能有幾種不同的原因。

解決方案 1：請嘗試下列步驟以診斷故障：

1. 請注意錯誤訊息的第二部分，以尋找更多資訊。跟進 You must use a valid fully-formed launch template 錯誤，參閱更具體的錯誤訊息，以識別您將需要解決的問題。

2. 如果找不到原因，請使用 [run-instances](#) 命令。使用 `--dry-run` 選項，如以下範例所示。這可讓您重現問題，並提供有關其原因的見解。

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --dry-run
```

3. 如果值無效，請驗證指定的資源是否存在以及它是否正確。例如，當您指定 Amazon EC2 金鑰對時，資源必須存在於您建立或更新 Auto Scaling 群組的帳戶和區域中。
4. 如果缺少預期的資訊，請確認您的設定並視需要調整啟動範本。
5. 進行變更後，請重新執行含有 `--dry-run` 選項的 [run-instances](#) 命令，以確認您的啟動範本是否使用有效值。

如需詳細資訊，請參閱 [為 Auto Scaling 群組建立啟動範本](#)。

您沒有使用啟動範本的授權 (許可不足)

問題：在嘗試為 Auto Scaling 群組指定啟動範本時，您收到 `You are not authorized to use launch template` 錯誤。

原因 1：如果您嘗試使用啟動範本，且您正使用的 IAM 憑證沒有足夠的權限，則您會收到錯誤，表示您無權使用啟動範本。

解決方案 1：若要解決問題，請嘗試下列動作：

- 確認您用來發出請求的 IAM 憑證擁有呼叫所需 EC2 API 動作 (包括 `ec2:RunInstances` 動作) 的權限。如果在啟動範本中指定了任何標籤，您還必須有使用 `ec2:CreateTags` 動作的許可。
- 或者，確認您用來發出請求的 IAM 憑證已指派 `AmazonEC2FullAccess` 政策。此 AWS 受管政策授予所有 Amazon EC2 資源和相關服務的完整存取權，包括 Amazon EC2 Auto Scaling、CloudWatch 和 Elastic Load Balancing。

如需使用啟動範本所需許可的詳細資訊，包括範例 IAM 政策，請參閱《Amazon EC2 使用者指南》中的 [使用 IAM 許可控制啟動範本的存取](#)。如需 IAM 政策的範例，請參閱 [在 Auto Scaling 群組中控制 Amazon EC2 啟動範本用量](#)。

原因 2：如果嘗試使用指定執行個體設定檔的啟動範本，則必須具有傳遞與執行個體設定檔相關聯 IAM 角色的 IAM 許可。

解決方案 2：確認您用來發出請求的 IAM 憑證擁有正確的 `iam:PassRole` 許可，可將指定角色傳遞給 Amazon EC2 Auto Scaling 服務。如需詳細資訊和 IAM 政策範例，請參閱 [在 Amazon EC2 執行個體上](#)

[執行的應用程式的 IAM 角色](#)。如需執行個體設定檔相關的進一步故障診斷主題，請參閱《IAM 使用者指南》中的 [Amazon EC2 和 IAM 故障診斷](#)。

原因 3：如果您嘗試使用在另一個中指定 AMI 的啟動範本 AWS 帳戶，且 AMI 是私有的，且不會與您 AWS 帳戶正在使用的共用，則會收到您無權使用啟動範本的錯誤。

解決方案 3：確認 AMI 的許可包含您正在使用的帳戶。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[與特定共用 AMI AWS 帳戶](#)。

相關資訊

以下相關資源可協助您使用此服務。

資源	Description
《 Amazon EC2 Auto Scaling API 參考 》	每個 API 操作的文件均會顯示請求參數和 XML 回應，並提供特定語言 SDK 參考主題的連結。
《AWS CLI 命令參考》中的 autoscaling	您可以使用 Auto Scaling 群組的 AWS CLI 命令描述。
AWS Tools for PowerShell Cmdlet 參考	Tools AWS for PowerShell 可讓您從 PowerShell 命令列編寫 AWS 資源操作的指令碼。
使用 建立 Auto Scaling 群組 AWS CloudFormation	AWS::AutoScaling::AutoScalingGroup 資源可讓您建置、建模並管理 Auto Scaling 群組，而不必手動採取行動。
AWS 一般參考 中的 Amazon EC2 Auto Scaling 端點和配額	Amazon EC2 Auto Scaling 區域和端點的相關資訊。
產品頁面	Amazon EC2 Auto Scaling 相關資訊的主要網頁。
AWS re : Post	AWS 受管問答 (Q & A) 服務，為您的技術問題提供群眾來源、專家審查的答案。
《Amazon EC2 使用者指南》中的 建立 AMI	了解如何從自訂執行個體建立 Amazon Machine Image (AMI)。
《Amazon EC2 使用者指南》中的 連接至 Linux 執行個體	了解如何連接至您啟動的 Linux 執行個體。
《Amazon EC2 使用者指南》中的 連線至 Windows 執行個體	了解如何連接至您啟動的 Windows 執行個體。
《Amazon CloudWatch 使用者指南》中的 建立帳單警示來監控預估的 AWS 費用	了解如何使用 CloudWatch 監控您的預估費用。

資源	Description
Application Auto Scaling 使用者指南	了解如何為 Amazon Web Services 以外的可擴展資源設定自動擴展Amazon EC2。

文件歷史紀錄

下表說明 Amazon EC2 Auto Scaling 文件自 2018 年 7 月起的重要增補。如需有關此文件更新的通知，您可以訂閱 RSS 摘要。

變更	描述	日期
安全 IAM 更新	AutoScalingServiceRolePolicy 現在包含 <code>ec2:CreateReplaceRootVolumeTask</code> 許可，可讓您在執行個體重新整理時取代執行個體上的根磁碟區。此外，已從政策中移除 <code>events:DescribeRule</code> 許可。	2025 年 11 月 20 日
取代根磁碟區	根磁碟區取代只會取代根 EBS 磁碟區，同時讓執行個體保持執行狀態，以更新您的執行個體。如需詳細資訊，請參閱在 執行個體重新整理期間取代根磁碟區 。	2025 年 11 月 20 日
執行個體生命週期政策	執行個體生命週期政策可讓您控制終止生命週期關聯未成功完成時會發生的情況。如需詳細資訊，請參閱在 執行個體重新整理期間取代根磁碟區 。	2025 年 11 月 20 日
同步佈建	您現在可以使用 <code>LaunchInstances</code> API，透過立即的意見回饋同步啟動執行個體，並精確控制在哪些可用區域執行個體中啟動。如需詳細資訊，請參閱 同步佈建 。	2025 年 11 月 18 日

[生命週期掛鉤的 Lambda 通知目標](#)

您現在可以直接將 lifecycle hook 通知路由至 AWS Lambda 函數。如需詳細資訊，請參閱[AWS Lambda 直接將通知路由到](#)。

2025 年 7 月 28 日

[高解析度指標](#)

目標追蹤現在支援高解析度 CloudWatch 指標，其中包含以低於一分鐘的間隔發佈的秒層級資料點。如需詳細資訊，請參閱[使用高解析度指標建立目標追蹤政策以加快回應速度](#)。

2024 年 11 月 22 日

[安全 IAM 更新](#)

[AutoScalingServiceRolePolicy](#) 受管政策現在會將其他許可授予資源群組 resource-groups:ListGroupResources 。

2024 年 11 月 20 日

[效能保護](#)

為 Auto Scaling 群組使用屬性型執行個體類型選擇時，您現在可以啟用效能保護，以確保選取的執行個體類型類似於或超過指定的效能基準。如需詳細資訊，請參閱[使用屬性型執行個體類型選擇建立混合執行個體群組](#)。

2024 年 11 月 20 日

[容量保留偏好設定](#)

您現在可以優先在容量保留中啟動執行個體。如需詳細資訊，請參閱[容量預留](#)。

2024 年 11 月 20 日

[區域轉移](#)

您現在可以使用區域轉移，從可用區域中的應用程式損害中復原。如需詳細資訊，請參閱[Auto Scaling 群組區域轉移](#)。

2024 年 11 月 18 日

[可用區域分佈](#)

您現在可以為您的 Auto Scaling 群組選擇可用區域分佈。如需詳細資訊，請參閱 [Auto Scaling 群組可用區域分佈](#)。

2024 年 11 月 7 日

[安全 IAM 更新](#)

[AutoScalingServiceRolePolicy](#) 受管政策現在會將其他許可授予 Amazon EC2 (ec2:GetSecurityGroupsForVpc 和 ec2:GetInstanceTypesFromInstanceRequirements)。

2024 年 2 月 29 日

[其他 支援暖集區休眠 AWS 區域](#)

您現在可以在兩個額外區域的暖集區中讓執行個體休眠：AWS GovCloud (美國東部) 和 AWS GovCloud (美國西部)。如需有關暖集區的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的暖集區](#)。

2024 年 2 月 26 日

[其他 支援暖集區休眠 AWS 區域](#)

您現在可以在兩個額外區域的暖集區中讓執行個體休眠：歐洲 (蘇黎世) 和中東 (阿拉伯聯合大公國)。如需有關暖集區的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的暖集區](#)。

2024 年 2 月 21 日

[支援跨帳戶參數使用](#)

您現在可以 AWS 帳戶 透過 Amazon EC2 Auto Scaling 使用從另一個 共用的 AWS Systems Manager 參數。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[在啟動範本中使用 AWS Systems Manager 參數而非 AMI IDs](#)。

2024 年 2 月 21 日

[新的 Spot 價格保護選項](#)

您現在可以使用屬性型執行個體類型選擇時，將 Spot 執行個體的價格保護閾值定義為隨需價格的百分比。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[價格保護](#)。

2024 年 1 月 29 日

[執行個體維護政策](#)

您現在可以使用執行個體維護政策，定義在導致執行個體被取代的事件 (包括執行個體重新整理) 發生期間，是否要在現有執行個體終止之前或之後啟動執行個體。如需更多詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[執行個體維護政策](#)一節。

2023 年 11 月 15 日

[ML 的容量區塊](#)

您現在可以在建立啟動範本時指定容量區塊保留 ID，以在容量區塊中啟動執行個體。透過容量區塊，您便可為日後保留 GPU 執行個體，以支援時間的機器學習 (ML) 工作負載。如需詳細資訊，請參閱「Amazon EC2 Auto Scaling 使用者指南」的[使用容量區塊處理機器學習工作負載](#)。

2023 年 10 月 31 日

[全新執行個體重新整理功能](#)

您現在可以設定執行個體重新整理，以便在其偵測到指定 CloudWatch 警示進入 ALARM 狀態時，將執行個體的狀態設為失敗並選擇是否要復原。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用復原功能來復原變更](#)。

2023 年 7 月 31 日

[指南變更](#)

已在指南中新增在容量保留中啟動隨需執行個體的新主題。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用隨需容量保留在特定可用區域中預留容量](#)。

2023 年 7 月 28 日

[指南變更](#)

有關將 AWS CloudFormation 堆疊從啟動組態遷移到啟動範本的新主題已新增至指南。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[從啟動組態將 AWS CloudFormation 堆疊遷移至啟動範本](#)一節。

2023 年 4 月 18 日

支援新的 API 操作

此版本新增了三項新的 API 操作，分別是 AttachTrafficSources、DetachTrafficSources 和 DescribeTrafficSources。此外，新欄位 TrafficSources 也新增至 DescribeAutoScalingGroups 操作的結果中。新活動狀態 WaitingForConnectionDraining 也新增至 DescribeScalingActivities 操作的結果中。Amazon EC2 Auto Scaling 也支援新的數值 VPC_LATTICE，適用於 CreateAutoScalingGroup、UpdateAutoScalingGroup 和 DescribeAutoScalingGroups 操作中的 HealthCheckType 欄位。如需詳細資訊，請參閱 [《Amazon EC2 Auto Scaling API 參考》](#)。

2023 年 3 月 31 日

支援 Amazon VPC Lattice

這是適用於 Amazon EC2 Auto Scaling 之 VPC Lattice 的正式發佈版本。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[將 Auto Scaling 群組流量路由至 VPC Lattice 目標群組](#)。

2023 年 3 月 31 日

[指南變更](#)

包含使用 Elastic Load Balancing AWS CLI 範例的 區段現在包含新的和更新的 範例。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用 Elastic Load Balancing 搭配 AWS Command Line Interface \(AWS CLI\) 的範例](#)。

2023 年 3 月 31 日

[支援其他 中的預測擴展 AWS 區域](#)

您現在可以在中東 (阿拉伯聯合大公國) 和 AWS GovCloud (美國東部) 區域建立預測擴展政策。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[Amazon EC2 Auto Scaling 的預測擴展](#)。

2023 年 3 月 16 日

[全新執行個體重新整理功能](#)

您現在可以選擇終止或忽略處於待命狀態的執行個體，並取代或忽略受縮減保護的執行個體，而不必等到它們變成可取代的狀態。您也可以從失敗的執行個體重新整理中復原變更。在此更新中，本文檔已擴充為納入下列主題：復原執行個體重新整理、取消執行個體重新整理，以及了解執行個體重新整理時可設定參數的預設值。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[根據執行個體重新整理來取代 Auto Scaling 執行個體](#)。

2023 年 2 月 10 日

[支援使用 AMI ID 的 AWS Systems Manager 參數](#)

您現在可在啟動範本中使用 Systems Manager 參數，而非 AMI ID。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[在啟動範本中使用 AWS Systems Manager 參數而非 AMI ID](#)。

2023 年 1 月 19 日

[預測擴展建議](#)

您現在可以從 Amazon EC2 Auto Scaling 主控台，取得評估和選擇正確預測擴展政策的建議。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[評估您的預測擴展政策](#)。

2023 年 1 月 18 日

[預測擴展預測](#)

預測擴展產生的預測現在每六小時更新一次，而不是每天更新一次。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[Amazon EC2 Auto Scaling 的預測擴展](#)。

2023 年 1 月 6 日

[CloudWatch 指標數學的支援](#)

建立目標追蹤擴展政策時，您現在可以使用指標數學。利用指標數學，可查詢多個 CloudWatch 指標，並使用數學表達式根據這些指標來建立新的時間序列。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用指標數學為 Amazon EC2 Auto Scaling 建立目標追蹤擴展政策](#)。

2022 年 12 月 8 日

[更新至 IAM 服務連結角色許可](#)

AutoScalingService RolePolicy 政策現向 Amazon EC2 Auto Scaling 授予額外許可。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的 AWS 受管政策](#)。

2022 年 12 月 6 日

[新的 Spot 分配策略](#)

您現在可以使用價格和容量最佳化分配策略來請求 Spot 集區中最不可能中斷且價格最低的 Spot 執行個體。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [分配策略](#)。

2022 年 11 月 10 日

[支援亞太區域 \(雅加達\) 的預測擴展](#)

您現在可以在亞太區域 (雅加達) 建立預測擴展政策。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的預測擴展](#)。

2022 年 10 月 13 日

[支援主控台中預測擴展的自訂指標](#)

現在您可以在從 Amazon EC2 Auto Scaling 主控台建立預測擴展政策時，使用自訂指標。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的預測擴展](#)。

2022 年 10 月 13 日

[用於預測擴展指標的 CloudWatch 監控](#)

您現在可以使用 CloudWatch 存取監控資料以進行預測擴展。這可讓您使用指標數學來建立顯示預測資料準確度的新時間序列。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用 CloudWatch 監控預測擴展指標](#)。

2022 年 7 月 7 日

[支援亞太區域 \(大阪\) 的預測擴展](#)

您現在可在亞太區域 (大阪) 建立預測擴展政策。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[Amazon EC2 Auto Scaling 的預測擴展](#)。

2022 年 7 月 6 日

[其他區域中支援的暖集區休眠](#)

您現在可在四個其他區域的暖集區中休眠執行個體：非洲 (開普敦)、亞太區域 (雅加達)、亞太區域 (大阪) 以及歐洲 (米蘭)。如需有關暖集區的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[Amazon EC2 Auto Scaling 的暖集區](#)。

2022 年 7 月 5 日

[更新至運作狀態檢查](#)

在執行運作狀態檢查時，Amazon EC2 Auto Scaling 現在可以幫助您最大限度地減少因臨時問題或設定錯誤的運作狀態檢查而可能導致的停機時間。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的「[Amazon EC2 Auto Scaling 如何將停機時間降至最低](#)」。

2022 年 5 月 21 日

[預設執行個體暖機](#)

您現在可以統一 Auto Scaling 群組的所有暖機和冷卻設定，並透過啟用預設執行個體暖機期來最佳化持續擴展政策的效能。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[設定 Auto Scaling 群組的預設執行個體暖機](#)。

2022 年 4 月 19 日

[指南變更](#)

有關與其他 AWS 服務整合的新章節已新增至指南。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[與 Amazon EC2 Auto Scaling 整合的AWS 服務](#)。

2022 年 3 月 29 日

[更新至 IAM 服務連結角色許可](#)

AutoScalingService RolePolicy 政策現向 Amazon EC2 Auto Scaling 授予額外讀取許可。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[Amazon EC2 Auto Scaling 的AWS 受管政策](#)。

2022 年 3 月 28 日

[執行個體中繼資料提供目標生命週期狀態](#)

您可以從執行個體中繼資料中擷取 Auto Scaling 執行個體的目標生命週期狀態。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[透過執行個體中繼資料擷取目標生命週期狀態](#)。

2022 年 3 月 24 日

[新暖集區功能支援](#)

現在，您可以在暖集區中讓執行個體進入休眠，即可停止執行個體，而不刪除其記憶體內容 (RAM)。現在，您還可以在縮減規模時將執行個體傳回暖集區，而不是一直終止稍後需要的執行個體容量。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的暖集區](#)。

2022 年 2 月 24 日

[指南變更](#)

Amazon EC2 Auto Scaling 主控台已更新其他選項，幫助您在啟用略過相符項目並指定所需組態的情況下，啟動執行個體重新整理。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [啟動或取消執行個體重新整理 \(主控台\)](#)。

2022 年 2 月 3 日

[用於預測擴展政策的自訂指標](#)

您現在可以選擇在建立預測擴展政策時是否要使用自訂指標。您還可以使用指標數學進一步自訂政策中包含的指標。如需詳細資訊，請參閱 [使用自訂指標進行進階預測擴展政策設定](#)。

2021 年 11 月 24 日

[新的隨需分配策略](#)

現在，建立使用混合執行個體政策的 Auto Scaling 群組時，您可以選擇是否根據價格啟動隨需執行個體（首先是最低價格的執行個體類型）。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[分配策略](#)。

2021 年 10 月 27 日

[屬性型執行個體類型選取範圍](#)

Amazon EC2 Auto Scaling 新增了對屬性型執行個體類型選取範圍的支援。您可以將執行個體需求表示為一組屬性，例如 vCPU、記憶體和儲存，而不是手動選擇執行個體類型。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用屬性型執行個體類型選取範圍建立 Auto Scaling 群組](#)。

2021 年 10 月 27 日

[支援依標籤篩選群組](#)

現在，在使用 `describe-auto-scaling-groups` 命令擷取有關 Auto Scaling 群組的資訊時，您可以使用標籤篩選條件篩選您的 Auto Scaling 群組。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[使用標籤篩選 Auto Scaling 群組](#)。

2021 年 10 月 14 日

[指南變更](#)

Amazon EC2 Auto Scaling 主控台已更新，可協助您使用 建立自訂終止政策 AWS Lambda。已相應地修正主控台文件。如需詳細資訊，請參閱[使用不同的終止政策 \(主控台\)](#)。

2021 年 10 月 14 日

[支援將啟動組態複製到啟動範本](#)

您現在可以從 Amazon EC2 Auto Scaling 主控台將 AWS 區域中的所有啟動組態複製到新的啟動範本。

2021 年 8 月 9 日

[擴充執行個體重新整理功能](#)

您現在可以在取代執行個體時加入更新 (例如啟動範本的新版本)，方法是將所需組態新增至 `start-instance-refresh` 命令。透過啟用略過相符項目，您可以略過替代已經具有所需組態的執行個體。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[根據執行個體重新整理來取代 Auto Scaling 執行個體](#)。

2021 年 8 月 5 日

[支援自訂終止政策](#)

您現在可以使用 建立自訂終止政策 AWS Lambda。如需詳細資訊，請參閱[使用 Lambda 建立自訂終止政策](#)。指定終止政策的說明文件已隨之更新。

2021 年 7 月 29 日

[指南變更](#)

Amazon EC2 Auto Scaling 主控台已更新並使用額外的功能強化，可協助您建立具有指定時區的排程動作。[排程擴展](#)的說明文件已作相應修訂。

2021 年 6 月 3 日

[啟動組態中的 gp3 磁碟區](#)

您現在可以在啟動組態的區塊型設備映射中指定 gp3 磁碟區。

2021 年 6 月 2 日

[支援預測擴展](#)

您現在可以利用預測擴展功能，主動擴展使用擴展政策的 Amazon EC2 Auto Scaling 群組。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的預測擴展](#)。藉由這個更新，[AutoScalingServiceRolePolicy](#) 受管政策現在加入了呼叫 `cloudwatch:GetMetricData` API 動作的許可。

2021 年 5 月 19 日

[指南變更](#)

您現在可以從 GitHub 存取 lifecycle hook 的範例範本。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling User lifecycle hook](#)。

2021 年 4 月 9 日

[支援暖集區](#)

您現在可以透過將暖集區新增至 Auto Scaling 群組，平衡首次開機時間較長的應用程式的效能 (將冷啟動次數減到最少) 與成本 (停止過度佈建執行個體容量)。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的暖集區](#)。

2021 年 4 月 8 日

[支援檢查點](#)

您現在可以將檢查點新增至執行個體重新整理功能之中，以便分階段取代執行個體，並在特定時間點對執行個體執行驗證。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[新增檢查點至執行個體重新整理](#)。

2021 年 3 月 18 日

[指南變更](#)

改進搭配 Amazon EC2 Auto Scaling 事件和 lifecycle hook 使用 EventBridge 的說明文件。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[搭配 EventBridge 使用 Amazon EC2 Auto Scaling](#) 和[教學課程：設定叫用 Lambda 函數的 lifecycle hook](#)。

2021 年 3 月 18 日

[支援本機時區](#)

您現在可以在本機時區中建立週期性排程動作，方法是將 `--time-zone` 選項新增至 `put-scheduled-update-group-action` 命令。如果您的時區遵循日光節約時間 (DST)，週期性動作會自動調整為日光節約時間。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[排程擴展](#)。

2021 年 3 月 9 日

擴充混合執行個體政策的功能

在使用混合執行個體政策時，您現在可針對 Spot 容量排定執行個體類型的優先順序。Amazon EC2 Auto Scaling 會竭盡全力嘗試履行個體類型優先順序，但會先針對容量最佳化。如需詳細資訊，請參閱「Amazon EC2 Auto Scaling 使用者指南」中的 [具備多個執行個體類型及購買選項的 Auto Scaling 群組](#)。

2021 年 3 月 8 日

已刪除群組的擴展活動

您現在可以檢視已刪除 Auto Scaling 群組的擴展活動，方法是將 `--include-deleted-groups` 選項新增至 `describe-scaling-activities` 命令。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的故障診斷](#)。

2021 年 2 月 23 日

主控台改進

您現在可從 Amazon EC2 Auto Scaling 主控台建立和連接 Application Load Balancer 或 Network Load Balancer。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [建立和連接新的 Application Load Balancer 或 Network Load Balancer \(主控台\)](#)。

2020 年 11 月 24 日

[多個網路介面](#)

您現在可以為指定多個網路介面的 Auto Scaling 群組設定啟動範本。如需詳細資訊，請參閱 [VPC 中的網路介面](#)。

2020 年 11 月 23 日

[多個啟動範本](#)

多個啟動範本現在可以搭配 Auto Scaling 群組使用。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [針對執行個體類型指定不同的啟動範本](#)。

2020 年 11 月 19 日

[閘道負載平衡器](#)

更新的指南說明如何將 Gateway Load Balancer 連接到 Auto Scaling 群組，確保 Amazon EC2 Auto Scaling 啟動的設備執行個體會自動從負載平衡器註冊或取消註冊。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Elastic Load Balancing 類型](#)和[將負載平衡器連接到 Auto Scaling 群組](#)。

2020 年 11 月 10 日

[執行個體生命週期上限](#)

您現在可將執行個體的生命週期上限減少為一天 (86,400 秒)。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [根據 Auto Scaling 執行個體生命週期上限來取代執行個體](#)。

2020 年 11 月 9 日

[容量重新平衡](#)

您可以設定 Auto Scaling 群組，在 Amazon EC2 發出重新平衡建議時啟動替代 Spot 執行個體。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 容量重新平衡](#)。

2020 年 11 月 4 日

[執行個體中繼資料服務第 2 版](#)

您可以請求使用執行個體中繼資料服務第 2 版，這是工作階段導向方法，可讓您在使用啟動組態時請求執行個體中繼資料。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [設定執行個體中繼資料選項](#)。

2020 年 7 月 28 日

[指南變更](#)

[控制規模縮減期間應終止的 Auto Scaling 執行個體](#)、[監控 Auto Scaling 執行個體和群組](#)、[啟動範本](#) 以及《Amazon EC2 Auto Scaling 使用者指南》中的 [啟動組態](#) 章節中的各種改進和新主控台程序。

2020 年 7 月 28 日

[執行個體重新整理](#)

在進行組態變更時，啟動執行個體重新整理來更新 Auto Scaling 群組中的所有執行個體。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [根據執行個體重新整理來取代 Auto Scaling 執行個體](#)。

2020 年 6 月 16 日

[指南變更](#)

[根據執行個體生命週期上限來取代 Auto Scaling 執行個體、具有多種執行個體類型和購買選項的 Auto Scaling 群組、根據 Amazon SQS 進行擴展](#)以及《Amazon EC2 Auto Scaling 使用者指南》中的[標記 Auto Scaling 群組和執行個體](#)章節中的各種改進。

2020 年 5 月 6 日

[指南變更](#)

IAM 文件的各種改進。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[啟動範本支援](#)和[Amazon EC2 Auto Scaling 身分型政策範例](#)。

2020 年 3 月 4 日

[停用擴展政策](#)

您現在可以停用和重新啟用擴展政策。這項功能可讓您在保留組態詳細資訊時，暫時停用擴展政策，以便之後能夠再次啟用該政策。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[停用 Auto Scaling 群組的擴展政策](#)。

2020 年 2 月 18 日

[新增通知功能](#)

當您的 Auto Scaling 群組因為缺少安全群組或啟動範本而無法向外擴展 AWS Health 儀板表時，Amazon EC2 Auto Scaling 現在會將事件傳送到您的。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[Amazon EC2 Auto Scaling 的 Health 儀板表通知](#)。

2020 年 2 月 12 日

指南變更

[Amazon EC2 Auto Scaling 與 IAM 搭配運作的方式](#)、[Amazon EC2 Auto Scaling 身分型政策範例](#)、[與加密磁碟區搭配使用時必要的 CMK 金鑰政策](#)以及《Amazon EC2 Auto Scaling 使用者指南》中的[監控 Auto Scaling 執行個體和群組](#)章節的各種改進和修正。

2020 年 2 月 10 日

指南變更

改進使用執行個體加權之 Auto Scaling 群組的文件。了解如何在使用「容量單位」測量所需容量時使用擴展政策。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[擴展政策的運作方式和擴展調整類型](#)。

2020 年 2 月 6 日

新增「安全」章節

《[Amazon EC2 Auto Scaling 使用者指南](#)》中新增的安全章節，有助於您了解如何在使用 Amazon EC2 Auto Scaling 時套用[共同的責任模型](#)。在此更新中，使用者指南章節「控制對 Amazon EC2 Auto Scaling 資源的存取」已替換為更有幫助的新章節 [Amazon EC2 Auto Scaling 的 Identity and Access Management](#)。

2020 年 2 月 4 日

[執行個體類型的建議](#)

AWS Compute Optimizer 提供 Amazon EC2 執行個體建議，協助您改善效能、節省成本或兩者。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[取得執行個體類型的建議](#)。

2019 年 12 月 3 日

[專用主機和主機資源群組](#)

已更新指南來顯示如何建立一個指定主機資源群組的啟動範本。這可讓您建立一個具有啟動範本的 Auto Scaling 群組，該範本會指定要在專用主機上使用的 BYOL AMI。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[建立 Auto Scaling 群組的啟動範本](#)。

2019 年 12 月 3 日

[支援 Amazon VPC 端點](#)

您現在可以在 VPC 和 Amazon EC2 Auto Scaling 之間建立私有連線。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[Amazon EC2 Auto Scaling 和介面 VPC 端點](#)。

2019 年 11 月 22 日

[執行個體生命週期上限](#)

您現在可以指定服務中執行個體的時間長度上限，將執行個體自動取代。如果有任何執行個體接近此限制，Amazon EC2 Auto Scaling 會逐漸取代它們。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[根據 Auto Scaling 執行個體生命週期上限來取代執行個體](#)。

2019 年 11 月 19 日

[執行個體權重](#)

對於具有多個執行個體類型的 Auto Scaling 群組，您現在可以選擇性地指定每個執行個體類型對群組容量貢獻的容量單位數量。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [Amazon EC2 Auto Scaling 的執行個體加權](#)。

2019 年 11 月 19 日

[執行個體類型數目下限](#)

您不再需要為 Spot、隨需和預留執行個體的群組指定額外的執行個體類型。對於所有 Auto Scaling 群組，現在的數目下限是一個執行個體類型。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 [具備多個執行個體類型及購買選項的 Auto Scaling 群組](#)。

2019 年 9 月 16 日

[支援新的 Spot 分配策略](#)

Amazon EC2 Auto Scaling 現在支援新的 Spot 配置策略「容量優化」，使用 Spot 執行個體集區依據可用 Spot 容量優化選擇以滿足您的請求。如需詳細資訊，請參閱「Amazon EC2 Auto Scaling 使用者指南」中的 [具備多個執行個體類型及購買選項的 Auto Scaling 群組](#)。

2019 年 8 月 12 日

[指南變更](#)

改善了[服務連結角色](#)和[與加密磁碟區搭配使用時必要的 CMK 金鑰政策](#)主題中的 Amazon EC2 Auto Scaling 文件。

2019 年 8 月 1 日

[支援標記增強功能](#)

Amazon EC2 Auto Scaling 現在會新增標籤至 Amazon EC2 執行個體，作為啟動執行個體同一 API 呼叫的一部分。如需詳細資訊，請參閱[標記 Auto Scaling 群組和執行個體](#)。

2019 年 7 月 26 日

[指南變更](#)

改善了[暫停及繼續擴展程序](#)主題中的 Amazon EC2 Auto Scaling 文件。已更新[客戶受管政策範例](#)，以包含一個範例政策，可讓使用者只將特定的自訂字尾服務連結角色傳遞至 Amazon EC2 Auto Scaling。

2019 年 6 月 13 日

[支援全新的 Amazon EBS 功能](#)

啟動範本主題新增全新 Amazon EBS 功能的支援。變更從快照還原時 EBS 磁碟區的加密狀態。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[建立 Auto Scaling 群組的啟動範本](#)。

2019 年 5 月 13 日

[指南變更](#)

在以下各章節中改進了 Amazon EC2 Auto Scaling 文件：[控制規模縮減期間應終止的 Auto Scaling 執行個體](#)、[Auto Scaling 群組](#)、[具有多種執行個體類型和購買選項的 Auto Scaling 群組](#)，以及[Amazon EC2 Auto Scaling 的動態擴展](#)。

2019 年 3 月 12 日

[結合執行個體類型和購買選項的支援](#)

在購買選項 (Spot、隨需和預留執行個體) 與單一 Auto Scaling 群組內的執行個體類型之間，佈建和自動擴展執行個體。如需詳細資訊，請參閱「Amazon EC2 Auto Scaling 使用者指南」中的[具備多個執行個體類型及購買選項的 Auto Scaling 群組](#)。

2018 年 11 月 13 日

[根據 Amazon SQS 進行擴展的更新主題](#)

更新的指南說明如何使用自訂指標來擴展 Auto Scaling 群組，以因應 Amazon SQS 佇列不斷改變的需求。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的[根據 Amazon SQS 進行擴展](#)。

2018 年 7 月 26 日

下表說明 Amazon EC2 Auto Scaling 文件在 2018 年 7 月前的重要變更。

功能	Description	發行日期
支援目標追蹤擴展政策	為您的應用程式設定動態擴展只需幾個步驟。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的目標追蹤擴展政策 。	2017 年 7 月 12 日
資源層級許可的支援	建立 IAM 政策以控制資源層級的存取。如需詳細資訊，請參閱 控制對 Amazon EC2 Auto Scaling 資源的存取 。	2017 年 5 月 15 日
監控改善	Auto Scaling 群組指標不再需要啟用詳細監控功能。您現在可以啟用群組指標集合，並從主控台的 Monitoring (監控) 索引標籤檢視指標圖。如需詳細資訊，請參閱 使用 Amazon CloudWatch 監控 Auto Scaling 群組和執行個體 。	2016 年 8 月 18 日

功能	Description	發行日期
支援 Application Load Balancer	將一或多個目標群組連接到新的或現有 Auto Scaling 群組。如需詳細資訊，請參閱 將負載平衡器連接到 Auto Scaling 群組 。	2016 年 8 月 11 日
lifecycle hook 事件	Amazon EC2 Auto Scaling 會在呼叫 lifecycle hook 時，將事件傳送至 EventBridge。如需詳細資訊，請參閱 在 Auto Scaling 群組擴展時取得 EventBridge 。	2016 年 2 月 24 日
執行個體保護	防止 Amazon EC2 Auto Scaling 在規模縮減時選擇終止特定執行個體。如需詳細資訊，請參閱 執行個體保護 。	2015 年 12 月 07 日
步進擴展政策	建立擴展政策，其可讓您依據警示違規的程度擴展。如需詳細資訊，請參閱 擴展政策類型 。	2015 年 7 月 06 日
更新負載平衡器	將負載平衡器連接到現有 Auto Scaling 群組，或從其上分離負載平衡器。如需詳細資訊，請參閱 將負載平衡器連接到 Auto Scaling 群組 。	2015 年 6 月 11 日
支援 ClassicLink	將 Auto Scaling 群組的 EC2-Classic 執行個體連結到 VPC，使連結的 EC2-Classic 執行個體與 VPC 中的執行個體使用私有 IP 地址通訊。如需詳細資訊，請參閱 將 EC2-Classic 執行個體連結至 VPC 。	2015 年 1 月 19 日
lifecycle hook	您在執行個體執行動作的同時，將剛啟動或終止中的執行個體保留在待處理狀態。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling lifecycle hook 。	2014 年 7 月 30 日
分離執行個體	將執行個體從 Auto Scaling 群組分離。如需詳細資訊，請參閱 從 Auto Scaling 群組分離 EC2 執行個體 。	2014 年 7 月 30 日
將執行個體置於待命狀態	將執行個體從 InService 狀態改為 Standby 狀態。如需詳細資訊，請參閱 從 Auto Scaling 群組暫時移除執行個體 。	2014 年 7 月 30 日
管理標籤	使用 AWS 管理主控台管理 Auto Scaling 群組。如需詳細資訊，請參閱 標記 Auto Scaling 群組和執行個體 。	2014 年 5 月 01 日

功能	Description	發行日期
支援專用執行個體	啟動專用執行個體，做法是當您建立啟動組態時指定置放租用屬性。如需詳細資訊，請參閱 執行個體安置租用 。	2014 年 4 月 23 日
從 EC2 執行個體建立群體或啟動組態	建立 Auto Scaling 群組或使用 EC2 執行個體啟動組態。如需有關使用 EC2 執行個體建立啟動組態的資訊，請參閱 使用 EC2 執行個體建立啟動組態 。如需有關使用 EC2 執行個體建立 Auto Scaling 群組的詳細資訊，請參閱 使用 EC2 執行個體建立 Auto Scaling 群組 。	2014 年 1 月 02 日
連接執行個體	將執行個體連接到現有 Auto Scaling 群組，藉此啟用適用於 EC2 執行個體的自動擴展。如需詳細資訊，請參閱 將 EC2 執行個體連接到 Auto Scaling 群組 。	2014 年 1 月 02 日
檢視帳戶限制	檢視您帳戶 Auto Scaling 資源的限制。如需詳細資訊，請參閱 Auto Scaling 資源和群組的配額 。	2014 年 1 月 02 日
主控台支援 Amazon EC2 Auto Scaling	使用 存取 Amazon EC2 Auto Scaling AWS 管理主控台。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 入門 。	2013 年 12 月 10 日
指派公有 IP 地址	指派公有 IP 地址給啟動至 VPC 的執行個體。如需詳細資訊，請參閱 在 VPC 中啟動 Auto Scaling 執行個體 。	2013 年 9 月 19 日
執行個體終止政策	為終止 EC2 執行個體時使用的 Amazon EC2 Auto Scaling 指定執行個體終止政策。如需詳細資訊，請參閱 控制規模縮減期間應終止的 Auto Scaling 執行個體 。	2012 年 9 月 17 日
支援 IAM 角色	使用 IAM 執行個體設定檔啟動 EC2 執行個體。您可以使用此功能來指派 IAM 角色到執行個體，讓您的應用程式安全地存取其他 Amazon Web Services。如需詳細資訊，請參閱 使用 IAM 角色啟動 Auto Scaling 執行個體 。	2012 年 6 月 11 日
支援 Spot 執行個體	啟動具有啟動組態的 Spot 執行個體。如需詳細資訊，請參閱 請求適用於容錯和彈性應用程式的 Spot 執行個體 。	2012 年 6 月 7 日

功能	Description	發行日期
標記 群組和執行個體	標記 Auto Scaling 群組，然後指定該標籤在建立後也適用於啟動的 EC2 執行個體。如需詳細資訊，請參閱 標記 Auto Scaling 群組和執行個體 。	2012 年 1 月 26 日
支援 Amazon SNS	只要 Amazon EC2 Auto Scaling 啟動或終止 EC2 執行個體，就可以使用 Amazon SNS 接收通知。如需詳細資訊，請參閱在 Auto Scaling 群組擴展時取得 SNS 通知。 Amazon EC2 Auto Scaling 也新增以下新功能： • 能夠使用 Cron 語法設定重複擴展活動。如需詳細資訊，請參閱PutScheduledUpdateGroupAction API 操作。 • 新的組態設定，可讓您不必將啟動的執行個體新增到負載平衡器 (LoadBalancer) 即能進行擴展。如需詳細資訊，請參閱ProcessType API 資料類型。 • DeleteAutoScalingGroup 操作中的 ForceDelete 旗標會指示 Amazon EC2 Auto Scaling 刪除具備相關聯執行個體的 Auto Scaling 群組，無須等待執行個體先終止。如需詳細資訊，請參閱DeleteAutoScalingGroup API 操作。	2011 年 7 月 20 日
排定的擴展動作	增加對排定的擴展動作的支援。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的排程擴展 。	2010 年 12 月 2 日
支援 Amazon VPC	新增對 Amazon VPC 的支援。如需詳細資訊，請參閱 在 VPC 中啟動 Auto Scaling 執行個體 。	2010 年 12 月 2 日
支援 HPC 叢集	新增對高效能運算 (HPC) 叢集的支援。	2010 年 12 月 2 日
支援運作狀態檢查	新增了搭配 Amazon EC2 Auto Scaling 受管 EC2 執行個體使用 Elastic Load Balancing 運作狀態檢查的支援。如需詳細資訊，請參閱 Auto Scaling 群組中執行個體的運作狀態檢查 。	2010 年 12 月 2 日

功能	Description	發行日期
支援 CloudWatch 警示	移除了舊的觸發機制並且重新設計了 Amazon EC2 Auto Scaling 來使用 CloudWatch 警示功能。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的動態擴展 。	2010 年 12 月 2 日
暫停和恢復擴展	新增支援至暫停和恢復擴展程序。	2010 年 12 月 2 日
支援 IAM	新增對 IAM 的支援。如需詳細資訊，請參閱 控制對 Amazon EC2 Auto Scaling 資源的存取 。	2010 年 12 月 2 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。