



託管指南

Amazon GameLift Servers



Amazon GameLift Servers: 託管指南

Copyright © 2025 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，這些擁有者可能隸屬於 Amazon，或與 Amazon 有合作關係，或由 Amazon 贊助。

Table of Contents

什麼是 Amazon GameLift Servers ?	1
您可以使用 做什麼 Amazon GameLift Servers	1
如何使用 Amazon GameLift Servers	1
Amazon GameLift Servers 解決方案	2
託管選項	2
FlexMatch 進行配對	6
FleetIQ 用於自我管理的 Amazon EC2 託管	6
Realtime 具有可自訂伺服器邏輯的伺服器	6
受管託管解決方案架構	7
使用 託管的遊戲元件	7
託管解決方案資源	8
Amazon GameLift Servers 的運作方式	9
託管遊戲伺服器	9
執行遊戲工作階段	10
擴展機群容量的規模	11
監控 Amazon GameLift Servers	11
使用其他 AWS 資源	12
容器的運作方式	12
容器機群元件	12
常見架構	14
核心功能	14
玩家如何連接到遊戲	18
服務位置	19
支援 AWS 的位置	19
受管託管的位置	23
Amazon GameLift ServersAnywhere 的位置	24
的位置 Amazon GameLift ServersFlexMatch	24
Amazon GameLift Servers 在中國	24
定價	24
產生定價預估	25
管理遊戲託管成本	28
開始使用	30
開始之前	30
快速加入選項	30

自訂開發選項	31
Amazon GameLift Servers 範例	31
自訂遊戲伺服器範例	31
Amazon GameLift ServersRealtime 範例	32
設定 AWS 帳戶	32
註冊 AWS 帳戶	33
建立具有管理存取權的使用者	33
設定 的使用者許可 Amazon GameLift Servers	34
為使用者設定程式設計存取	35
為您的遊戲設定程式設計存取	36
IAM 許可範例	37
設定 IAM 服務角色	41
取得開發工具	45
對於遊戲伺服器	45
對於遊戲用戶端服務	48
用於資源管理	48
對於 Amazon GameLift ServersRealtime	49
教學課程：使用Amazon GameLift Servers包裝函式快速入門	49
先決條件	50
概觀	50
步驟 1：取得並建置遊戲伺服器包裝函式	51
步驟 2：準備您的遊戲伺服器建置	51
步驟 3：設定機群的包裝函式	52
步驟 4：上傳遊戲伺服器組建	53
步驟 5：建立受管 EC2 機群	54
步驟 6：建立並連線至遊戲工作階段	55
步驟 7：管理和監控您的機群	56
步驟 8：擴展您的遊戲伺服器	56
對常見問題進行故障診斷	57
後續步驟	59
使用遊戲引擎外掛程式探索	59
外掛程式工作流程	60
Unreal Engine 的外掛程式	60
Unity 的外掛程式 (伺服器 SDK 5.x)	89
Unity 外掛程式 (伺服器 SDK 4.x)	107
受管 EC2 開發藍圖	129

受管容器開發藍圖	134
Anywhere 開發藍圖	139
混合式託管開發藍圖	144
準備 的遊戲 Amazon GameLift Servers	151
將遊戲與自訂遊戲伺服器整合	151
Amazon GameLift Servers 互動	152
整合遊戲伺服器	155
整合遊戲用戶端	163
遊戲引擎和 Amazon GameLift Servers	169
設計您的後端服務	192
驗證您的玩家	193
無伺服器後端	193
WebSocket 型後端	194
設定反覆開發	195
建置以雲端為基礎的測試環境	196
設定本機測試	199
使用 代理程式	203
設定本機測試 (舊版)	205
設定Amazon GameLift Servers本機	206
測試遊戲伺服器	206
測試遊戲伺服器和用戶端	209
本機 的變化	212
新增FlexMatch配對	212
取得機群資料	213
將遊戲與 整合 Amazon GameLift ServersRealtime	214
什麼是Realtime伺服器？	214
管理遊戲工作階段	214
用戶端伺服器互動	214
自訂伺服器	215
部署和更新	216
整合遊戲用戶端	216
自訂Realtime指令碼	222
部署遊戲伺服器軟體	228
部署自訂伺服器建置	228
部署受管託管	228
部署 for Anywhere 託管	229

封裝遊戲建置檔案	229
新增建置安裝指令碼	230
建立受管託管的建置	232
更新受管託管的遊戲伺服器建置	236
建置容器映像	237
建立遊戲伺服器容器映像	238
將容器映像推送至 Amazon ECR	240
部署Amazon GameLift ServersRealtime指令碼	241
套件指令碼檔案	242
從本機目錄上傳指令碼檔案	242
從 Amazon S3 上傳指令碼檔案	243
更新指令碼檔案	245
設定託管機群	246
機群特性	246
機群建立的運作方式	247
受管 EC2 機群	247
受管 EC2 機群建立工作流程	248
建立受管 EC2 機群	249
自訂您的 EC2 受管機群	257
更新機群組態	262
偵錯機群問題	265
遠端連線至機群執行個體	268
VPC 對等互連	276
受管容器機群	281
建立容器機群	283
更新容器機群	286
自訂容器機群	288
建立容器群組定義	295
更新容器群組定義	299
刪除容器群組定義	301
Anywhere 機群	303
Anywhere 機群建立工作流程	304
建立 Anywhere 機群	304
使用別名抽象機群	312
建立別名	313
編輯別名	313

使用佇列管理遊戲工作階段放置	315
佇列特性	315
建立佇列	316
自訂佇列	319
最佳實務	319
定義佇列的範圍	320
建置多位置佇列	321
排定遊戲工作階段置放的優先順序	322
評估佇列指標	326
Spot 執行個體的設計	327
設定事件通知	329
設定 SNS 主題	329
使用伺服器端加密設定 SNS 主題	331
設定 EventBridge	332
教學課程：使用 Spot 執行個體建立佇列	333
步驟 1：定義佇列的範圍	334
步驟 2：建立 Spot 機群基礎設施	334
步驟 3：為每個機群指派別名	334
步驟 4：建立具有目的地的佇列	335
步驟 5：將延遲限制新增至佇列	335
Summary	336
擴展託管容量	337
在 主控台中管理機群容量	337
設定託管容量限制	338
設定容量限制	338
手動設定機群容量	339
暫停自動擴展	339
手動設定機群容量	340
自動擴展機群容量	341
目標型自動擴展	341
規則型自動擴展	343
擴展容器機群	347
準備遊戲啟動	349
準備就緒	349
準備進行測試	349
準備啟動	350

推出後計劃	350
使用 管理託管資源 AWS CloudFormation	351
最佳實務	351
使用 AWS CloudFormation 堆疊	352
單一位置的堆疊	352
多個區域的堆疊	352
更新組建	353
自動部署建置更新	353
手動部署建置更新	354
轉返的運作方式	354
監控 Amazon GameLift Servers	355
指標比較	355
依來源的指標可用性	355
選擇正確的監控來源	358
主題	358
在主控台中追蹤遊戲託管	358
託管儀表板	359
遊戲伺服器組建	360
Amazon GameLift Servers指令碼Realtime	361
機群	362
機群詳細資訊	363
遊戲工作階段和玩家工作階段	366
Aliases	370
遊戲工作階段佇列	371
使用 CloudWatch 監控	373
指標維度	374
機群指標	375
Container Fleet 指標	385
佇列指標	389
FlexMatch 指標	392
FleetIQ 指標	395
記錄 API 呼叫	397
CloudTrail 中的 Amazon GameLift Servers 資訊	397
了解 Amazon GameLift Servers 日誌檔案項目	398
記錄伺服器訊息	400
自訂伺服器的記錄	401

記錄 Amazon GameLift ServersRealtime	403
安全	408
資料保護	408
靜態加密	410
傳輸中加密	410
網際網路流量隱私權	411
身分與存取管理	411
目標對象	411
使用身分驗證	412
使用政策管理存取權	414
Amazon GameLift Servers 搭配 IAM 的運作方式	416
身分型政策範例	423
疑難排解	428
AWS 受管政策	430
使用 記錄和監控 Amazon GameLift Servers	431
法規遵循驗證	432
恢復能力	433
基礎設施安全性	433
組態與漏洞分析	434
安全最佳實務	434
讓機群執行時間環境保持在最新狀態	435
保護您的連接埠組態	436
其他安全資源	437
參考指南	438
服務 API (AWS SDK)	438
管理Amazon GameLift Servers託管資源	439
開始遊戲工作階段並加入玩家	442
伺服器 SDK 5.x	443
伺服器 SDK 5.x 更新	443
遷移至伺服器 SDK 5.x	445
C++ 伺服器開發套件	446
C# 伺服器 SDK	491
Go 伺服器開發套件	530
C++ (非真實) 伺服器 SDK：動作	555
伺服器 SDK 4 和更早版本	598
適用於 C++ 的伺服器 SDK：動作	598

適用於 C# 的伺服器 SDK：動作	620
適用於 Unreal 的伺服器 SDK：動作	640
Amazon GameLift Servers 參考 Realtime	655
即時用戶端 API (C#) 參考	655
Realtime 指令碼參考	668
遊戲工作階段置放事件	676
置放事件語法	676
PlacementFulfilled	677
PlacementCancelled	679
PlacementTimedOut	680
PlacementFailed	681
AMI 版本	682
服務端點和配額	683
Amazon GameLift Servers API 限制	683
API 限制參考	684
資源層級限流	692
Amazon GameLift Servers SDK API 限制	692
UDP ping 信標	695
UDP ping 信標的運作方式	695
UDP ping 信標的常見使用案例	696
取得信標端點	696
實作延遲測量	697
程式碼範例	698
版本備註和 SDK 版本	706
軟體開發套件版本	706
版本備註	727
.....	dcclxv

什麼是 Amazon GameLift Servers ？

使用 Amazon GameLift Servers 在雲端部署、操作和擴展以工作階段為基礎的多玩家遊戲專用、低成本伺服器。以 AWS 全球運算基礎設施為基礎，Amazon GameLift Servers 有助於提供高效能、高可靠性的遊戲伺服器，同時動態擴展您的資源使用量以滿足全球玩家需求。

您可以使用 做什麼 Amazon GameLift Servers

Amazon GameLift Servers 支援這些使用案例等：

- 使用 Amazon GameLift Servers 受管 EC2 託管，在雲端託管您自己的自訂多玩家遊戲伺服器。
- 使用 [Amazon Elastic Compute Cloud \(Amazon EC2\)](#) Spot 執行個體執行低成本受管託管資源。
- 託管您的容器化遊戲伺服器，以跨平台實現靈活性，並支援使用 Amazon GameLift Servers 受管容器進行遷移。
- 建立混合託管解決方案，以支援多雲端和/或內部部署託管，同時使用 Amazon GameLift Servers Anywhere 在同一個位置管理遊戲工作階段。
- 使用 Amazon GameLift Servers 為您的多玩家遊戲建立強大的配對系統 FlexMatch。
- 自動擴展受管託管容量，以根據實際玩家用量滿足您的遊戲需求。
- 使用在單一位置管理您的 Amazon EC2 Amazon GameLift Servers 運算資源以進行遊戲 FleetIQ。
- 使用 Amazon GameLift Servers Anywhere 和 EC2 為您的遊戲伺服器和用戶端建置建立反覆測試環境。
- 對於不需要自訂建置遊戲伺服器的遊戲，請使用設定輕量型伺服器解決方案 Amazon GameLift Servers Realtime。

Tip

[立即開始使用 Amazon GameLift Servers](#)

如何使用 Amazon GameLift Servers

使用下列工具搭配 Amazon GameLift Servers。

AWS CLI

使用 AWS Command Line Interface (AWS CLI) 呼叫 AWS SDK，包括的服務 API Amazon GameLift Servers。請參閱 AWS Command Line Interface 《使用者指南》中的 [入門 AWS CLI](#)。

Amazon GameLift Servers 主控台

使用 [AWS Management Console 適用於 Amazon GameLift Servers](#) 的來設定資源、管理您的遊戲伺服器部署，以及追蹤效能和用量指標。Amazon GameLift Servers 主控台是以程式設計方式或使用管理資源的 GUI 替代方案 AWS CLI。

Amazon GameLift Servers SDK

Amazon GameLift Servers SDKs 包含在遊戲用戶端、遊戲伺服器、遊戲服務與 Amazon GameLift Servers 服務之間建立通訊所需的程式庫。如需詳細資訊，請參閱 [取得 Amazon GameLift Servers 開發工具](#)。

適用於的用戶端 SDK Amazon GameLift ServersRealtime

適用於的用戶端 SDK Amazon GameLift ServersRealtime 可讓您將遊戲用戶端連線至提供的 Realtime 伺服器 Amazon GameLift Servers、加入遊戲工作階段，以及與其他玩家保持同步。下載 [SDK](#) 並進一步了解如何使用 [Amazon GameLift ServersRealtime 用戶端 API \(C#\) 進行 API](#) 呼叫。

Amazon GameLift Servers 解決方案

Amazon GameLift Servers 為正在建置工作階段型多玩家遊戲的開發人員提供各種解決方案。

遊戲開發人員的解決方案

- [Amazon GameLift Servers 託管選項](#)
- [Amazon GameLift ServersFlexMatch 進行配對](#)
- [Amazon GameLift ServersFleetIQ 用於自我管理的 Amazon EC2 託管](#)
- [Amazon GameLift ServersRealtime 搭配可自訂的伺服器邏輯](#)

Amazon GameLift Servers 託管選項

使用 Amazon GameLift Servers 操作遊戲伺服器時，您有數個選項，可讓您的遊戲伺服器託管位置和方式。無論您想要使用已經擁有的託管資源，還是想要設定由管理的雲端型託管 Amazon GameLift Servers，都可以為玩家建立無縫的託管體驗。

[受管 EC2](#)

[受管容器](#)

[混合託管](#)

[隨處託管](#)

受管 EC2

透過Amazon GameLift Servers受管 EC2 託管，您可以卸載管理遊戲伺服器的大部分工作。從各種 Amazon EC2 執行個體類型中選擇運算資源。整合您的遊戲專案，讓 Amazon GameLift Servers處理詳細資訊。如需受管託管的詳細資訊，請參閱 [Amazon GameLift Servers 的運作方式](#)。

[開始為您的遊戲開發Amazon GameLift Servers受管託管解決方案。](#)

主要功能

- 託管在 Amazon Linux 或 Windows Server 作業系統上執行的多玩家遊戲。
- 無論玩家身在何處，都能提供低延遲的遊戲體驗。在 Amazon GameLift Servers 支援的任何 AWS 區域 和 Local Zones 全球部署遊戲伺服器。如需完整清單，請參閱 [Amazon GameLift Servers 服務位置](#)。
- 使用Amazon GameLift Servers智慧型遊戲工作階段配置，讓玩家永遠獲得最佳的託管玩家體驗。您可以依賴Amazon GameLift Servers決策，也可以根據置放條件自訂，例如成本、玩家延遲和地理位置。
- 選擇如何擴展您的託管資源以滿足玩家需求。手動管理容量或設定自動擴展。透過目標型自動擴展，您可以維持動態大小的閒置容量緩衝區，這可協助您控制成本，同時確保新玩家可以在最短的等待時間進入遊戲。
- 讓 Amazon GameLift Servers 部署和管理以雲端為基礎的遊戲伺服器。會視需要Amazon GameLift Servers建立資源、安裝遊戲伺服器軟體，並自動啟動為玩家託管遊戲工作階段的程序。設定自訂運作狀態追蹤，並讓 Amazon GameLift Servers偵測和解決效能不佳的資源。
- 利用Amazon GameLift Servers監控功能來評估效能和用量。您可以追蹤硬體效能、遊戲工作階段置放效率和伺服器程序生命週期等因素的指標。您可以追蹤作用中的遊戲工作階段和玩家工作階段，以觀察一段時間內的用量。您也可以下載和存放遊戲工作階段日誌。
- 對於生產託管，請使用 和 的 AWS CloudFormation 範本自動化您的遊戲託管資源管理和部署 Amazon GameLift Servers AWS Cloud Development Kit (AWS CDK)。利用持續整合和持續交付 (CI/CD) 工具和服務，例如 AWS CodePipeline。

受管容器

Amazon GameLift Servers 為容器化遊戲伺服器提供完整的雲端託管解決方案。透過Amazon GameLift Servers受管容器，您可以利用容器使用的核心優勢，例如可攜性、敏捷性和容錯能力。受管容器機群可使用下列功能。

[開始為您的容器化遊戲伺服器開發Amazon GameLift Servers受管託管解決方案。](#)

主要功能

- 開發具有輕量型容器的自訂架構，以在 Amazon GameLift Servers Linux 型託管資源上執行遊戲伺服器軟體。
- 使用 Docker 工具建立以 Linux 為基礎的容器映像。將要部署的映像儲存在 Amazon Elastic Container Registry (Amazon ECR) 儲存庫中。
- 將容器機群資源部署到 Amazon GameLift Servers 支援的任何 AWS 區域 或 Local Zone，以提供低延遲玩家體驗。請參閱 [Amazon GameLift Servers 服務位置](#)。
- 使用工具管理機群生命週期，以建立遊戲伺服器版本的模型並部署機群更新。
- 使用Amazon GameLift Servers遊戲工作階段置放功能，包括佇列和FlexMatch配對，為您的玩家尋找最佳的遊戲工作階段配對。
- 使用 Anywhere 機群，使用 Amazon GameLift Servers服務測試您的遊戲伺服器和容器架構。在本機或在雲端測試環境中測試您的遊戲。
- 使用容器特定的效能指標追蹤遊戲託管效能。使用硬體指標監控機群資源的運作狀態。
- 使用的 AWS CloudFormation 範本管理容器機群資源Amazon GameLift Servers。

混合託管

使用 Amazon GameLift Servers服務搭配Amazon GameLift Servers受管託管和 Anywhere 自我管理託管的組合。混合式方法可讓您建置現在所需的解決方案，同時為未來需要達到的目標做好準備。混合式解決方案有意義的常見案例包括：

- 將您的託管解決方案擴展至 AWS 雲端。透過新增Amazon GameLift Servers受管託管，補充現有託管解決方案（內部部署硬體或其他雲端託管）的功能。透過受管託管，您可以增加託管容量或新增「爆量」容量，以快速擴展並僅在需要時支付資源。您也可以利用Amazon GameLift Servers服務的全球足跡，觸及世界各地更多玩家，並提供他們預期的低延遲多玩家體驗。
- 準備遷移至雲端型託管。如果您正在考慮或計劃遷移到 AWS 雲端（而不是升級您自己的硬體），混合式託管解決方案是您逐步進行轉換的可行方式。

- 提升 服務位置以外的玩家延遲Amazon GameLift Servers。如果您已經在使用Amazon GameLift Servers受管託管，您可能需要在某些情況下支援玩家。例如，您可能想要接觸異常偏遠位置的玩家，或大幅降低這些區域的延遲。新增自訂託管位置，並使用 Amazon GameLift Servers Anywhere 來管理這些位置以及您的受管託管資源。

開始為您的遊戲開發Amazon GameLift Servers混合託管解決方案。

主要功能

- 將相同的遊戲用戶端和伺服器元件與受管和自我管理的託管資源搭配使用。在所有託管資源中提供統一的玩家體驗。
- 使用相同的FlexMatch配對建構器，在所有託管資源中放置配對。
- 當您將混合託管資源部署到全球各地時，可集中管理您的混合託管資源。
- 隨著玩家需求波動，在受管和自我管理的資源之間順暢地管理遊戲工作階段負載。
- 透過 Amazon GameLift Servers代理程式，您可以使用相同的工具來管理所有類型託管資源的遊戲伺服器生命週期。
- 收集所有託管資源的遊戲和玩家指標和日誌。利用 Amazon GameLift Servers 功能和其他 AWS 服務來合併資料，並開發具凝聚力的可觀測性解決方案。

隨處託管

使用具有Amazon GameLift Servers遊戲工作階段管理的 Amazon GameLift ServersAnywhere 機群，包括配對，以隨心所欲地託管自訂遊戲伺服器。Anywhere機群特別適用於測試環境，以進行快速、反覆的遊戲開發。為您自己的本機工作站或一組雲端託管資源設定機Anywhere群。對於生產託管，您可以針對由Amazon GameLift Servers受管Anywhere機群補充的內部部署硬體，使用混合方法搭配機群。

如需使用 Anywhere 測試的詳細資訊，請參閱 [使用 Amazon GameLift ServersAnywhere 設定本機測試](#)。如需設定 Anywhere 機群的詳細資訊，請參閱 [使用 設定託管機群 Amazon GameLift Servers](#)。

開始為您的遊戲開發 Amazon GameLift Servers Anywhere 託管解決方案。

主要功能

- 在您開發多玩家遊戲時，執行快速的反覆測試。
- 使用Amazon GameLift Servers工具來管理在您自己的硬體上託管的遊戲伺服器。
- 利用任何位置最接近玩家的可用硬體。

Amazon GameLift ServersFlexMatch 進行配對

使用 FlexMatch建置自訂規則集，為您的遊戲定義多玩家配對。FlexMatch使用規則集來比較每個配對的相容玩家，並為玩家提供理想的多玩家體驗。

如需的詳細資訊FlexMatch，請參閱[什麼是Amazon GameLift ServersFlexMatch？](#)

主要功能

- 平衡比對建立速度和比對品質。
- 根據定義的特性配對玩家或隊伍。
- 定義規則，根據延遲將玩家放入配對。

Amazon GameLift ServersFleetIQ 用於自我管理的 Amazon EC2 託管

使用FleetIQ遊戲伺服器群組直接使用 Amazon EC2 和 Amazon EC2 Auto Scaling 中的託管資源。這可為低成本、彈性的遊戲託管提供Amazon GameLift Servers最佳化的優勢。此解決方案適用於需要比全受管Amazon GameLift Servers解決方案更彈性的遊戲開發人員。

如需有關 如何FleetIQ搭配 Amazon EC2 和 EC2 Auto Scaling 進行遊戲託管的資訊，請參閱 [Amazon GameLift ServersFleetIQ開發人員指南](#)。

主要功能

- 使用FleetIQ演算法取得最佳化 Spot 執行個體平衡。
- 使用玩家路由功能有效率地管理您的遊戲伺服器資源，並提供更好的玩家加入遊戲體驗。
- 根據玩家用量自動擴展託管容量。
- 直接管理您自己的 Amazon EC2 執行個體 AWS 帳戶。
- 使用任何支援的遊戲伺服器可執行檔格式，包括 Windows、Linux、容器和 Kubernetes。

Amazon GameLift ServersRealtime 搭配可自訂的伺服器邏輯

使用Realtime伺服器託管不需要自訂建置遊戲伺服器的遊戲。此輕量型伺服器解決方案提供遊戲伺服器，您可以針對遊戲進行設定。您可以使用 Amazon GameLift Servers受管託管解決方案來託管 Realtime伺服器。

如需使用 託管的詳細資訊Amazon GameLift ServersRealtime，請參閱 [將遊戲與 整合 Amazon GameLift ServersRealtime](#)。

主要功能

- 使用Amazon GameLift Servers管理功能，包括自動擴展、多位置佇列和遊戲工作階段放置。
- 使用Amazon GameLift Servers託管資源，並選擇機群的 AWS 運算硬體類型。
- 利用完整的網路堆疊進行遊戲用戶端和伺服器互動。
- 取得伺服器邏輯可自訂的遊戲伺服器核心功能。
- 對 Realtime 組態和伺服器邏輯進行即時更新。

受管Amazon GameLift Servers解決方案架構

本主題中的圖表概述 Amazon GameLift Servers 的完整託管解決方案的結構。

使用 託管的遊戲元件

下圖說明受管Amazon GameLift Servers託管解決方案的關鍵元件如何一起運作，以執行專用遊戲伺服器，並協助玩家尋找和連線至託管遊戲工作階段。您為遊戲開發的託管解決方案將包含大多數或所有這些元件。

此架構的關鍵元件包括下列項目：

遊戲用戶端

遊戲用戶端是您在玩家裝置上執行的軟體。玩家透過在託管遊戲伺服器上加入遊戲工作階段來玩您的遊戲。遊戲用戶端要求透過後端服務加入遊戲工作階段、接收遊戲工作階段的連線資訊，並使用它直接與遊戲工作階段連線。如需詳細資訊，請參閱[準備的遊戲 Amazon GameLift Servers](#)。連線至 Realtime 伺服器時，遊戲用戶端會使用用戶端 SDK Amazon GameLift ServersRealtime。

後端服務

後端服務是您建立的自訂服務，可Amazon GameLift Servers代表遊戲用戶端處理與服務的通訊。您也可以將後端服務用於遊戲特定的任務，例如玩家身分驗證和授權、庫存或貨幣控制。後端服務 Amazon GameLift Servers會使用 AWS SDK 中的 API 操作與服務通訊。

後端服務會提出請求，以取得現有的遊戲工作階段資訊並啟動遊戲工作階段。請求新遊戲工作階段會定義特定特性，例如玩家數量上限。這些請求會提示 Amazon GameLift Servers 啟動遊戲工作階段置放程序。當遊戲工作階段準備好接受玩家時，後端服務會擷取連線資訊，並將其提供給遊戲用戶端。

外部服務

您的遊戲可以依賴外部服務，例如驗證訂閱成員資格。外部服務可以透過後端服務和 將資訊傳遞到您的遊戲伺服器Amazon GameLift Servers。

遊戲伺服器

遊戲伺服器是遊戲的伺服器軟體，可在一組託管資源上執行。您可以將遊戲伺服器軟體上傳至 Amazon GameLift Servers，這會將其部署至託管資源並開始執行伺服器程序。每個遊戲伺服器程序都會與服務連線Amazon GameLift Servers，以表示準備託管遊戲工作階段。它與服務互動，以啟動遊戲工作階段、驗證新連線的玩家，並報告遊戲工作階段和玩家連線的狀態。

自訂遊戲伺服器Amazon GameLift Servers使用適用於 的伺服器 SDK 與 通訊Amazon GameLift Servers。如需詳細資訊，請參閱 [將遊戲與自訂遊戲伺服器整合](#)。Realtime 伺服器是由 提供的遊戲伺服器Amazon GameLift Servers。您可以透過提供自訂指令碼來自訂伺服器邏輯。如需詳細資訊，請參閱[將遊戲與 整合 Amazon GameLift ServersRealtime](#)。

主機管理工具

設定和管理託管資源時，遊戲擁有者會使用託管管理工具來管理遊戲伺服器建置或指令碼、機群、配對和佇列。AWS 開發套件和 主控台內的Amazon GameLift Servers工具集提供多種方法來管理託管資源。您可以遠端存取任何個別遊戲伺服器進行故障診斷。

託管解決方案資源

下圖說明組成受管託管解決方案Amazon GameLift Servers的資源。提供自訂伺服器建置或Amazon GameLift ServersRealtime指令碼、部署運算機群以託管遊戲伺服器，然後設定遊戲工作階段佇列以尋找可用的託管資源並啟動新的遊戲工作階段。對於使用FlexMatch配對的遊戲，請新增配對組態和配對規則集來產生玩家配對。

遊戲伺服器程式碼

- 建置 – 您的自訂遊戲伺服器軟體，可在 上執行Amazon GameLift Servers，並為玩家託管遊戲工作階段。遊戲組建代表在特定作業系統上執行遊戲伺服器的一組檔案，您必須與 整合Amazon GameLift Servers。在您計劃設定機群 AWS 區域 的 Amazon GameLift Servers中，將遊戲組建檔案上傳至 。如需詳細資訊，請參閱[部署用於Amazon GameLift Servers託管的自訂伺服器建置](#)。
- 指令碼 – 用於 的組態和自訂遊戲邏輯Amazon GameLift ServersRealtime。Amazon GameLift ServersRealtime 為您的遊戲用戶端設定 ，方法是使用 JavaScript 建立指令碼，並新增自訂遊

戲邏輯來為玩家託管遊戲工作階段。如需詳細資訊，請參閱[部署 的指令碼 Amazon GameLift ServersRealtime](#)。

機群

運算資源的集合，可為您的玩家執行遊戲伺服器 and 託管遊戲工作階段。如需部署機群之位置的相關資訊，請參閱 [Amazon GameLift Servers 服務位置](#)。如需建立機群的詳細資訊，請參閱 [使用 設定 託管機群 Amazon GameLift Servers](#)。

Alias (別名)

機群的抽象識別符，可用來隨時變更玩家所連線的機群。如需詳細資訊，請參閱[建立Amazon GameLift Servers別名](#)。

遊戲工作階段佇列

一種遊戲工作階段置放機制，可接收新遊戲工作階段的請求，並搜尋可用的遊戲伺服器來託管新的工作階段。如需遊戲工作階段佇列的詳細資訊，請參閱[使用Amazon GameLift Servers佇列管理遊戲工作階段置放](#)。

Amazon GameLift Servers 的運作方式

本主題說明如何Amazon GameLift Servers管理多玩家遊戲伺服器的專用託管，並將其提供給玩家。它概述了核心功能的運作方式。

託管遊戲伺服器

使用 Amazon GameLift Servers，您可以透過數種不同的方式託管遊戲伺服器：受管 Amazon GameLift ServersAmazon GameLift ServersFleetIQ、和 Amazon GameLift ServersAnywhere。如需詳細資訊Amazon GameLift ServersFleetIQ，請參閱[什麼是Amazon GameLift ServersFleetIQ？](#)

您可以依照自己的遊戲需求設計機群。如需設計機群的詳細資訊，請參閱 [自訂您的 Amazon GameLift Servers EC2 受管機群](#)。

受管 Amazon GameLift Servers

透過 受管 Amazon GameLift Servers，您可以在稱為執行個體的Amazon GameLift Servers虛擬運算資源上託管遊戲伺服器。透過建立執行個體機群並部署它們來執行遊戲伺服器，來設定您的託管資源。

Amazon GameLift Servers隨處

透過 Amazon GameLift ServersAnywhere，您可以在管理的運算上託管遊戲伺服器。透過建立參考運算的 Anywhere 機群來設定您的託管資源。

機群別名

別名是在機群之間轉移的指定項目，使其成為擁有一般機群位置的便利方式。您可以使用別名將遊戲用戶端從使用一個機群切換到另一個機群，而無需變更您的遊戲用戶端。您也可以建立指向內容的終端機別名。

執行遊戲工作階段

將遊戲伺服器組建部署到機群並在每個執行個體上Amazon GameLift Servers啟動遊戲伺服器程序後，機群可以託管遊戲工作階段。當遊戲用戶端服務將置放請求傳送至後端服務或時，會Amazon GameLift Servers啟動新的遊戲工作階段Amazon GameLift Servers。

遊戲工作階段置放和FleetIQ演算法

佇列使用FleetIQ演算法來選取可用的遊戲伺服器，以託管新的遊戲工作階段。遊戲工作階段放置的關鍵元件是Amazon GameLift Servers遊戲工作階段佇列。您可以為遊戲工作階段佇列指派機群清單，以決定佇列可以放置遊戲工作階段的位置。如需遊戲工作階段佇列以及如何為您的遊戲設計它們的詳細資訊，請參閱 [自訂遊戲工作階段佇列](#)。

使用 UDP ping 信標最佳化遊戲工作階段放置

使用 Amazon GameLift Servers UDP ping 信標，您可以計算不同位置玩家和遊戲伺服器之間 UDP 封包的往返延遲，以協助挑選遊戲工作階段的最佳位置。如需 UDP ping 信標以及如何使用它們來測量延遲的詳細資訊，請參閱 [UDP ping 信標](#)。

- 對於使用遊戲工作階段佇列進行置放的遊戲，置放請求可以包含延遲資料，佇列會自動使用此資料來排定置放位置的優先順序。您可以進一步自訂優先順序，也可以設定包括延遲值上限的政策。請參閱 [排定遊戲工作階段置放的優先順序](#)。
- 對於未使用遊戲工作階段佇列但機群具有多個位置的遊戲，您可以在向 提出遊戲工作階段請求之前評估延遲資料並選擇最佳可用位置Amazon GameLift Servers。請參閱 中的取得和建立遊戲工作階段一節[Amazon GameLift Servers 新增至您的遊戲用戶端](#)。
- 如果您使用 FlexMatch 進行配對，您可以將配對規則設定為使用延遲資料。請參閱 Amazon GameLift Servers FlexMatch 開發人員指南中的[玩家請求配對](#)和[規則類型的延遲](#) FlexMatch規則區段。

玩家與遊戲的連線

作為遊戲工作階段放置程序的一部分，佇列或遊戲工作階段會提示選取的遊戲伺服器啟動新的遊戲工作階段。遊戲伺服器會回應提示，並在準備好接受玩家連線Amazon GameLift Servers時回報給。

Amazon GameLift Servers 然後將連線資訊交付給後端服務或遊戲用戶端服務。您的遊戲用戶端會使用此資訊直接連線至遊戲工作階段並開始遊戲。

擴展機群容量的規模

當機群處於作用中狀態並準備好託管遊戲工作階段時，您可以調整機群容量以滿足玩家需求。建議您在所有傳入玩家之間找到平衡，以快速找到遊戲，並過度花費在閒置的資源上。

Amazon GameLift Servers 提供高效的自動擴展工具，您也可以手動設定機群容量。如需詳細資訊，請參閱[使用 擴展遊戲託管容量 Amazon GameLift Servers](#)。

自動擴展

Amazon GameLift Servers 提供兩種自動擴展的方法：

- [目標型自動擴展](#)
- [使用規則型政策自動擴展](#)

其他擴展功能

- 遊戲工作階段保護 – Amazon GameLift Servers防止在縮減規模事件期間結束託管作用中玩家的遊戲工作階段。
- 擴展限制 – 透過設定機群中執行個體數量的下限和上限來控制整體執行個體用量。
- 暫停自動擴展 – 在機群位置層級暫停自動擴展，而不變更或刪除自動擴展政策。
- 擴展指標 – 追蹤機群的容量和擴展事件歷史記錄。

監控 Amazon GameLift Servers

當您啟動並執行機群時，會Amazon GameLift Servers收集各種資訊，協助您監控已部署遊戲伺服器的效能。您可以使用此資訊來最佳化資源的使用、疑難排解問題，並深入了解玩家在遊戲中的活動狀態。會Amazon GameLift Servers收集下列項目：

- 機群、位置、遊戲工作階段和玩家工作階段詳細資訊
- 用量指標
- 伺服器程序運作狀態
- 遊戲工作階段日誌

如需在 中監控的詳細資訊Amazon GameLift Servers，請參閱 [監控 Amazon GameLift Servers](#)。

使用其他 AWS 資源

您的遊戲伺服器 and 應用程式可以與其他 AWS 資源通訊。例如，您可以使用一組 Web 服務進行玩家身分驗證或社交聯網。若要讓您的遊戲伺服器存取您的 AWS 帳戶 管理 AWS 的資源，請明確允許 Amazon GameLift Servers 存取您的 AWS 資源。

Amazon GameLift Servers 提供管理這種存取類型的數個選項。如需詳細資訊，請參閱[與機群中的其他 AWS 資源通訊](#)。

容器在 中的運作方式 Amazon GameLift Servers

Amazon GameLift Servers 容器機群旨在讓您靈活地部署和擴展容器化應用程式。它使用 Amazon Elastic Container Service (Amazon ECS) 來管理機Amazon GameLift Servers群的任務部署和執行。本主題說明在Amazon GameLift Servers受管機群上執行容器的基本結構元素、說明常見架構，並概述一些核心概念。

 使用這些受管容器的工具加速入門：

- [容器入門套件](#)可簡化整合和機群設定。它會為您的遊戲伺服器新增必要的遊戲工作階段管理功能，並使用預先設定的範本，為您的遊戲伺服器建置容器機群和自動化部署管道。部署之後，請使用 Amazon GameLift Servers主控台和 API 工具來監控機群效能、管理遊戲工作階段和分析指標。
- 對於 Unreal Engine 或 Unity 開發人員，請使用[Amazon GameLift Servers外掛程式](#)來整合遊戲伺服器，並從遊戲引擎的開發環境中建置容器機群。外掛程式的引導式工作流程可協助您建立快速、簡單的解決方案，並使用受管容器進行雲端託管。然後在此基礎上建置，為您的遊戲建立自訂託管解決方案。

容器機群元件

機群

容器機群是 Amazon EC2 執行個體的集合，用於託管您的容器化遊戲伺服器。這些執行個體是由 Amazon GameLift Servers 代表您管理。當您建立機群時，您可以設定 容器架構與遊戲伺服器軟體如何部署到每個機群執行個體。您可以在一或多個地理位置中建立具有執行個體的容器機群。您

可以使用Amazon GameLift Servers擴展工具自動擴展容器機群的容量，以託管遊戲工作階段和玩家。

執行個體

Amazon EC2 執行個體是為遊戲託管提供運算容量的虛擬伺服器。使用 Amazon GameLift Servers，您可以從各種執行個體類型中進行選擇。每種執行個體類型都提供不同的 CPU、記憶體、儲存和聯網容量組合。

當您建立容器機群時，Amazon GameLift Servers 會根據您選擇的執行個體類型和機群組態來部署容器。每個部署的機群執行個體都是相同的，並以相同的方式執行容器化遊戲伺服器軟體。機群中的執行個體數量決定機群的大小和遊戲託管容量。

容器群組

Amazon GameLift Servers 使用容器群組的概念來描述和管理一組容器。容器群組類似於容器「任務」或「Pod」。在每個容器群組中，您可以定義容器的行為方式、設定相依性，以及共用可用的 CPU 和記憶體資源。

每個機群執行個體可以有下類型的容器群組：

- 遊戲伺服器容器群組會管理執行遊戲伺服器應用程式和支援軟體的容器。容器機群必須具有其中一種容器群組，才能託管遊戲工作階段和玩家。遊戲伺服器容器群組可以跨機群執行個體複寫。每個機群執行個體的遊戲伺服器群組複本數量取決於軟體的運算需求，以及執行個體上可用的運算資源。
- 每個執行個體容器群組是選用的，可讓您在每個機群執行個體上執行其他軟體。它們適用於執行背景服務或公用程式，例如用於監控。您的遊戲伺服器軟體不會直接依賴每個執行個體群組中的程序。每個機群執行個體只會部署一個執行個體容器群組的副本。

容器機群中的每個容器群組都有一個指定為「必要」的容器。基本容器可驅動容器群組的生命週期。如果基本容器失敗，整個容器群組會重新啟動。

容器

容器是容器型架構中最基本的元素。它包含具有軟體可執行檔和相依檔案的容器映像。定義容器，以設定軟體如何執行並與 互動Amazon GameLift Servers。

Amazon GameLift Servers 定義兩種類型的容器：

- 遊戲伺服器容器包含執行遊戲伺服器程序和為玩家託管遊戲工作階段所需的一切。它包含您的遊戲伺服器建置和相依的軟體。為機群的遊戲伺服器容器群組定義一個遊戲伺服器容器。遊戲伺服器容器會自動被視為對容器群組至關重要。

- 支援容器會執行其他軟體來支援您的遊戲伺服器。它類似於「附屬」容器的概念。它可讓您選擇搭配遊戲伺服器執行和擴展支援軟體，但管理為個別容器。在遊戲伺服器容器群組中，您可以定義零或多個支援容器。在每個執行個體容器群組中，所有容器都是支援容器。任何支援容器都可以指定為必要。

運算

運算代表機群執行個體上遊戲伺服器容器群組的複本。

常見架構

下圖說明最簡單的容器機群結構。在此結構中，機群中的每個執行個體都會維護一個遊戲伺服器容器群組的副本。容器群組具有執行一個遊戲伺服器程序的單一遊戲伺服器容器。在此範例中，容器機群設定為為每個執行個體放置一個遊戲伺服器容器群組的副本。使用此架構，每個執行個體都會執行一個遊戲伺服器程序。

第二個圖表說明更複雜的容器機群架構。在此結構中，機群同時具有遊戲伺服器容器群組和每個執行個體容器群組。遊戲伺服器容器群組具有遊戲伺服器程序和支援程序的個別容器。機群已設定為在每個機群執行個體上放置三個遊戲伺服器容器群組的副本。每個執行個體容器群組永遠不會複寫。在此範例中，容器機群設定為為每個執行個體放置三個遊戲伺服器容器群組的副本。使用此架構，每個執行個體都會執行三個遊戲伺服器程序。

核心功能

本節摘要說明 如何Amazon GameLift Servers實作一些基本容器概念。如需如何使用容器機群的指示，請參閱本指南中的相關主題。

作用中機群更新

受管容器提供進階支援，協助您管理託管軟體和容器架構的生命週期。您可以更新容器定義，包括容器映像，並將變更部署到現有的機群。此功能可讓您更快速、更輕鬆地在開發期間反覆執行容器的變更。它還提供可協助您建置、部署和追蹤一段時間內軟體版本更新的功能。這些功能包括：

- 管理容器群組定義更新和版本控制。您可以更新容器群組定義的幾乎所有屬性，包括容器映像和組態設定。每當您更新容器時，Amazon GameLift Servers會自動指派版本編號給更新，且預設會維護所有版本。您可以存取任何特定版本，也可以視需要刪除版本。建立容器機群時，您可以指定要部署的容器群組定義和版本。

- 使用新的容器群組定義和組態設定更新現有的容器機群。您可以將容器更新部署到已部署到機群執行個體的機群。您可以使用 AWS Management Console 或 AWS SDK 和 CLI，追蹤每個機群位置的更新部署狀態。
- 設定您希望機群更新在作用中機群之間部署的方式。
 - 遊戲工作階段保護。選擇在遊戲工作階段結束（安全部署）之前，使用作用中的遊戲工作階段保護機群執行個體。或者，選擇取代機群執行個體，無論遊戲工作階段活動為何（不安全部署）。在開發和測試階段使用不安全的部署，以減少部署時間。
 - 運作狀態百分比下限。指定您要在部署期間維護的正常運作任務百分比。此功能可讓您決定部署期間受影響的機群執行個體數量。低值會優先考慮部署速度，而高值則可確保在整個部署期間維持較高的遊戲伺服器可用性。
 - 部署失敗策略。決定部署失敗時要採取的動作。部署失敗表示某些更新的容器狀態檢查失敗，並被視為受損。您可以設定部署，將所有機群執行個體自動復原至先前部署的狀態。或者，您可以選擇維護一些受損的機群執行個體，以用於偵錯。

當您想要將更新部署到遊戲伺服器軟體時，更新作用中機群的功能非常有用。在您為遊戲伺服器建立新的容器映像之後，部署它是一個兩步驟的程序：首先，使用新映像更新容器群組定義，再接著更新容器機群。會視需要 Amazon GameLift Servers 處理所有其他任務。

容器包裝

開發容器結構以在容器機群中部署時，常見的目標是最佳化您使用的可用運算能力。為了達成此目標，您想要將盡可能多的遊戲伺服器容器群組封裝到每個機群執行個體。

Amazon GameLift Servers 根據下列資訊，計算每個執行個體的遊戲伺服器容器群組上限，協助您達成此目的：

- 機群的執行個體類型及其 vCPU 和記憶體資源。
- 遊戲伺服器容器群組中所有容器的 vCPU 和記憶體需求。

每個執行個體容器群組中所有容器的 vCPU 和記憶體需求，如果有的話。

建立容器機群時，您可以使用計算的最大值，也可以指定所需的數字。最佳實務是計劃試驗容器化遊戲伺服器軟體，以判斷最佳遊戲伺服器效能的資源需求。

容量擴展

機群容量會測量機群可以同時託管的遊戲工作階段數量。您也可以根據機群可支援的並行玩家數量來測量容量。若要增加或減少機群的託管容量，您可以新增或移除機群執行個體。

容器機群設定為在每個機群執行個體上執行特定數量的並行遊戲伺服器程序。（您可以根據 (1) 每個執行個體的遊戲伺服器容器群組，以及 (2) 每個容器群組中執行的遊戲伺服器程序數目來計算。）每個執行個體的 `cncurrent` 遊戲伺服器數量會告訴您新增或移除每個機群執行個體的影響。例如，如果您的容器機群在每個遊戲伺服器容器群組中執行 1 個遊戲伺服器程序，而且每個機群執行個體都擁有 100 個遊戲伺服器容器群組，則您可以增加或減少機群託管並行遊戲工作階段的容量，增量為 100。如果每個遊戲工作階段都有 10 個玩家位置，則您以 1000 的增量增加或減少機群託管玩家的容量。

透過容器機群，您可以使用提供的任何容量擴展方法 Amazon GameLift Servers。其中包含：

- 透過設定所需的機群執行個體計數，手動設定機群容量。
- 透過鎖定可用執行個體的所需緩衝區（目標追蹤）來設定自動擴展。此方法會自動維護一定數量的閒置託管資源，以便傳入的玩家可以快速進入遊戲。隨著玩家需求增加或減少，會持續調整此緩衝區的大小。
- 使用自訂擴展規則設定自動擴展（進階功能）。此方法可讓您根據您選擇的機群指標進行擴展。

遊戲用戶端/伺服器連線

透過 Amazon GameLift Servers 受管機群，遊戲用戶端會直接連線至雲端託管的遊戲伺服器。當遊戲用戶端要求加入遊戲時，會 Amazon GameLift Servers 尋找遊戲工作階段，並向遊戲用戶端提供連線資訊 (IP 和連接埠)。您可以為機群開啟特定連接埠範圍（傳入許可），以控制機群執行個體的外部存取。傳入許可會決定哪些連接埠開放給傳入流量。您可以快速關閉所有連接埠、限制為幾個連接埠，或開啟所有連接埠。

受管容器機群需要額外的設定，允許存取在容器中執行的程序。當您建立容器定義時，您可以指定一組連接埠，每個接受連線的程序各一個連接埠。其中包含：

- 將在遊戲伺服器容器中同時執行的所有遊戲伺服器程序。所有遊戲伺服器程序都必須允許遊戲用戶端連線，才能加入遊戲工作階段。
- 支援容器中任何外部來源需要連線的程序。例如，您可以遠端連線至測試應用程式。

當您設定面向內部的容器連接埠設定時，Amazon GameLift Servers 會使用它們來計算遊戲用戶端和其他應用程式可以連線的對外傳入許可。Amazon GameLift Servers 也會管理傳入許可與個別容器連接埠之間的映射，讓玩家能夠存取容器中的遊戲工作階段。此內部映射透過保護您的遊戲伺服器免於直接

存取容器連接埠，提供一層安全性。您可以選擇視需要自訂機群的對外連接埠設定。如需手動設定容器機群連接埠的詳細資訊，請參閱 [設定網路連線](#)。

您可以隨時修改容器機群的連接埠設定。此變更需要機群更新部署。

下圖說明跨容器機群的連接埠連線角色。如圖所示，您在個別容器上設定連接埠，Amazon GameLift Servers 並使用此資訊在機群執行個體上設定足夠的連接埠，以對應至每個容器連接埠。除非您選擇手動設定，否則外部執行個體傳入許可和內部連線連接埠都會由 Amazon GameLift Servers 為您的機群計算。

容器記錄

在受管容器機群中，會擷取所有容器的標準輸出（和標準錯誤）串流。這包括遊戲伺服器的遊戲工作階段日誌。您可以設定容器機群使用下列其中一個選項來處理輸出串流：

- 將容器輸出儲存為 Amazon CloudWatch 日誌串流。每個日誌串流都會參考機群 ID 和容器。如果您為機群選擇此記錄選項，您可以指定 CloudWatch 日誌群組，以組織機群中的所有日誌串流。然後，您可以視需要使用 CloudWatch 功能來搜尋和分析日誌資料。
- 將容器輸出儲存至 Amazon Simple Storage Service (Amazon S3) 儲存貯體。您可以視需要檢視、共用或下載內容。
- 關閉記錄。在此案例中，容器輸出不會儲存。

Amazon GameLift Servers 會將日誌資料從受管容器機群傳送至您 AWS 帳戶中的 CloudWatch 或 Amazon S3 服務。若要檢視您的資料，請登入 AWS 您的帳戶並使用個別服務，以使用 AWS Management Console 或其他工具。您可以透過為容器機群建立服務角色，將有限的存取權延伸至 Amazon GameLift Servers 以採取這些動作。

您可以隨時修改容器機群的記錄組態。此變更需要機群更新部署。

容器機群和 Amazon GameLift Servers 代理程式

常用的容器架構會在每個容器中執行單一程序。在 Amazon GameLift Servers 容器機群中，遊戲伺服器容器群組有一個遊戲伺服器容器，其會執行一個遊戲伺服器程序。使用此架構時，會 Amazon GameLift Servers 管理機群執行個體上每個遊戲伺服器容器群組中單一遊戲伺服器程序的生命週期。

如果您選擇建置容器架構，在每個遊戲伺服器容器群組中執行多個遊戲伺服器程序，您需要一種方法來管理所有程序的生命週期。這包括任務，例如視需要啟動、關閉和取代程序、管理要同時執行的所需程序數量，以及處理失敗狀態。

您可以選擇對這些任務使用 Amazon GameLift Servers 代理程式。對於容器機群，代理程式會實作執行時間指示，指定要執行的可執行檔（以及數量）、提供啟動參數，以及設定遊戲伺服器啟用的相關規則。例如，執行時間指示可能會指示代理程式維護十個遊戲伺服器程序以供生產使用，以及一個遊戲伺服器程序搭配特殊啟動參數用於測試。

若要將代理程式與容器機群搭配使用，請將代理程式新增至容器映像，並包含一組執行時間指示。如需代理程式的詳細資訊，請參閱 [使用 Amazon GameLift Servers 代理程式](#)。

玩家如何連接到遊戲

遊戲工作階段是在 Amazon GameLift Servers 上執行的遊戲執行個體。若要玩遊戲，玩家可以尋找並加入現有的遊戲工作階段，或建立新的遊戲工作階段並加入。玩家透過為遊戲工作階段建立玩家工作階段來加入。如果遊戲工作階段開放給玩家，則會為玩家 Amazon GameLift Servers 保留位置並提供連線資訊。玩家接著可以連接到遊戲工作階段並宣告保留的位置。

如需使用自訂遊戲伺服器建立和管理遊戲工作階段和玩家工作階段的詳細資訊，請參閱 [Amazon GameLift Servers 新增至您的遊戲用戶端](#)。如需將玩家連線至的資訊 Amazon GameLift ServersRealtime，請參閱 [整合的遊戲用戶端 Amazon GameLift ServersRealtime](#)。

Amazon GameLift Servers 提供與遊戲和玩家工作階段相關的多種功能。

在多個位置的最佳可用資源上託管遊戲工作階段

在設定 Amazon GameLift Servers 如何選取資源以託管新遊戲工作階段時，可從多個選項中進行選擇。如果您在多個位置執行機群，則可以設計遊戲工作階段佇列，在任何機群上放置新的遊戲工作階段，無論位置為何。使用 UDP ping 信標來計算延遲資料，佇列會自動使用此資料來排定置放位置的優先順序。

控制玩家對遊戲工作階段的存取

設定遊戲工作階段以允許或拒絕來自新玩家的加入請求，無論連線的玩家數量為何。

使用自訂遊戲和玩家資料

將自訂資料新增至遊戲工作階段物件和玩家工作階段物件。會在啟動新的遊戲工作階段時，將遊戲工作階段資料 Amazon GameLift Servers 傳遞至遊戲伺服器。當玩家連線到遊戲工作階段時，會將玩家工作階段資料 Amazon GameLift Servers 傳遞至遊戲伺服器。

篩選和排序可用的遊戲工作階段

使用工作階段搜尋和排序來尋找潛在玩家的最佳可能配對，或讓玩家從可用的遊戲工作階段清單中選擇。使用工作階段搜尋和排序，根據工作階段特性尋找遊戲工作階段。您也可以根據自己的自訂遊戲資料進行搜尋和排序。

追蹤遊戲和玩家用量資料

自動儲存已完成遊戲工作階段的日誌。您可以在整合Amazon GameLift Servers到遊戲伺服器時設定日誌儲存。如需詳細資訊，請參閱[在中記錄伺服器訊息 Amazon GameLift Servers](#)。

使用Amazon GameLift Servers主控台檢視遊戲工作階段的詳細資訊，包括工作階段中繼資料、設定和玩家工作階段資料。如需詳細資訊，請參閱[Amazon GameLift Servers 主控台](#)中的[遊戲和玩家工作階段](#)及[指標](#)。

Amazon GameLift Servers 服務位置

Amazon GameLift Servers 功能可在多個 AWS 區域 和 Local Zones 中使用。您可以設計託管解決方案，將遊戲伺服器放在玩家所在的位置。

支援 AWS 的位置

下表是支援 Amazon GameLift Servers 資源的 AWS 區域 和 Local Zones 清單。它指出您可以在每個位置建立的資源類型。

地理位置	位置碼	受管機群的主區域 (單一位置)	受管機群的主區域 (多位置)	受管機群的遠端位置 (多位置)	Anywhere 機群	遊戲工作階段佇列	FlexMatch 配對建構器和規則集
美國東部 (維吉尼亞北部)	us-east-1	是	是	是	是	是	是
美國東部 (俄亥俄)	us-east-2	是		是	是	是	
美國西部 (加利佛尼亞北部)	us-west-1	是		是	是	是	
美國西部 (奧勒岡)	us-west-2	是	是	是	是	是	是
Africa (Cape Town)	af-south-1			是			

地理位置	位置碼	受管機群的主區域 (單一位置)	受管機群的主區域 (多位置)	受管機群的遠端位置 (多位置)	Anywhere 機群	遊戲工作階段佇列	FlexMatch 配對建構器和規則集
亞太區域 (泰國)	ap-southeast-7			是			
亞太區域 (香港)	ap-east-1			是			
亞太地區 (馬來西亞)	ap-southeast-5			是			
亞太區域 (孟買)	ap-south-1	是		是	是	是	
亞太區域 (大阪)	ap-northeast-3			是			
亞太區域 (首爾)	ap-northeast-2	是	是	是	是	是	是
亞太區域 (新加坡)	ap-southeast-1	是		是	是	是	
亞太區域 (雪梨)	ap-southeast-2	是	是	是	是	是	是
亞太區域 (東京)	ap-northeast-1	是	是	是	是	是	是

地理位置	位置碼	受管機群的主區域 (單一位置)	受管機群的主區域 (多位置)	受管機群的遠端位置 (多位置)	Anywhere 機群	遊戲工作階段佇列	FlexMatch 配對建構器和規則集
加拿大 (中部)	ca-central-1	是		是	是	是	
歐洲 (法蘭克福)	eu-central-1	是	是	是	是	是	是
歐洲 (愛爾蘭)	eu-west-1	是	是	是	是	是	是
歐洲 (倫敦)	eu-west-2	是		是	是	是	
歐洲 (米蘭)	eu-south-1			是			
歐洲 (巴黎)	eu-west-3			是			
Europe (Stockholm)	eu-north-1			是			
Middle East (Bahrain)	me-south-1			是			
南美洲 (聖保羅)	sa-east-1	是		是	是	是	
亞特蘭大本地區域	us-east-1-atl-1			是			
芝加哥本地區域	us-east-1-chi-1			是			

地理位置	位置碼	受管機群的主區域 (單一位置)	受管機群的主區域 (多位置)	受管機群的遠端位置 (多位置)	Anywhere 機群	遊戲工作階段佇列	FlexMatch 配對建構器和規則集
達拉斯本地區域	us-east-1-dfw-1			是			
丹佛本地區域	us-west-2-den-1			是			
休士頓本地區域	us-east-1-iah-1			是			
堪薩斯市本地區域	us-east-1-mci-1			是			
洛杉磯本地區域	us-west-2-lax-1			是			
Phoenix 本機區域	us-west-2-phx-1			是			
奈及利亞拉哥斯當地區域	af-south-1-los-1			是			

Note

根據預設，並非所有 AWS 區域 都會啟用 AWS 帳戶。如果您想要在這些區域中具有執行個體的多位置機群，則必須啟用它們。如需未預設啟用的區域以及如何啟用它們的詳細資訊，請參閱《》中的[管理 AWS 區域](#) AWS 一般參考。您在 2022 年 2 月 28 日之前建立的機群不受影響。

此外，您必須更新Amazon GameLift Servers管理員政策以允許 `ec2:DescribeRegions` 動作。如需 區域預設未啟用的政策範例，請參閱 [管理許可範例](#)。

受管託管的位置

Amazon GameLift Servers 受管託管會部署遊戲伺服器資源的機群。每個機群都會在 中建立 AWS 區域，這是機群的主區域。機群的主區域會在機群的 Amazon Resource Number (ARN) 中參考。

您可以部署單一區域機群，僅在主區域中使用託管資源。或者，您可以部署多位置機群，並在多個地理位置託管資源。多位置機群具有主要區域和一或多個遠端位置。管理多位置機群的託管容量時，您可以個別設定每個位置的容量。

多位置機群的遠端位置可以是其他 AWS 區域 或 Local Zones。Local Zone 是 的延伸 AWS 區域。它可讓您將運算資源放在更接近使用者的位置，以提供低延遲的遊戲體驗。如需詳細資訊，請參閱 [AWS Local Zones](#)。Local Zone 的位置碼是其父區域碼，後面接著實體位置識別符。例如，洛杉磯本地區域的程式碼為 `us-west-2-lax-1`。

下圖說明在兩個 AWS 區域 和一個 Local Zone 中具有資源的多位置機群。機群的主區域是 `us-west-2`，而且有兩個遠端位置：`us-east-2`區域和`us-west-2-lax-1`本機區域。

除了機群資源之外，使用的受管託管Amazon GameLift Servers也可以使用下列類型的資源。您可以在 AWS 區域 支援 資源類型的特定 中建立這些資源。

- 組建 – 這是使用受管 EC2 機群託管的遊戲伺服器組建。在與要部署的機群相同的區域中建立建置資源。
- 指令碼 – 這是用來託管遊戲的組態指令碼Amazon GameLift ServersRealtime。在與要部署的機群相同的區域中建立指令碼資源。
- 容器群組定義和容器映像 – 這是在受管容器機群上執行容器的組態。它會使用要部署到容器機群的軟體來識別一或多個容器映像。建立容器群組定義和所有容器映像（存放在 Amazon Elastic Container Registry 儲存庫中），與要部署的機群位於相同區域中。
- 遊戲工作階段佇列 – 此資源會處理遊戲工作階段的請求，並啟動新的遊戲工作階段。處理會在 AWS 區域 佇列所在的 中進行。若要減少遊戲工作階段置放程序的延遲，請在要使用它的玩家附近，以地理位置建立佇列。

Amazon GameLift ServersAnywhere 的位置

Amazon GameLift ServersAnywhere 機群是您提供的託管硬體集合。您可以管理託管資源上的所有活動，包括部署遊戲伺服器軟體、保持更新，以及啟動伺服器程序。您可以建立 Anywhere 機群，將 Amazon GameLift Servers 服務與自我管理的託管資源連線。會 Amazon GameLift Servers 管理遊戲工作階段置放 - 處理玩家加入請求、尋找可用的託管資源、啟動新的遊戲工作階段，以及提供遊戲用戶端連線資訊。您可以在支援它們的任何 AWS 區域 中建立 Anywhere 機群。

您可以透過註冊將託管硬體的執行個體新增至 Anywhere 機群。每個已註冊的執行個體都必須有一個與其相關聯的自訂位置。自訂位置與 AWS 區域 或 Local Zones 無關。它們用於表示硬體的實體位置。

如需建立 Anywhere 機群和測試遊戲伺服器整合的詳細資訊，請參閱 [建立 Amazon GameLift ServersAnywhere 機群](#) 和 [使用 Amazon GameLift ServersAnywhere 設定本機測試](#)。

的位置 Amazon GameLift ServersFlexMatch

FlexMatch 資源用於處理玩家配對請求。其中包括配對組態資源和規則集資源。處理會在 AWS 區域 FlexMatch 資源所在的 中進行。若要減少配對過程中的延遲，請在接近將使用它的玩家地理位置上建立資源。配對組態及其使用的規則集必須位於相同的 中 AWS 區域。您可以在支援資源的任何 中建立 FlexMatch 資源 AWS 區域 。

如需 FlexMatch 為託管解決方案設定 的詳細資訊，請參閱 [Amazon GameLift ServersFlexMatch 開發人員指南](#)。

Amazon GameLift Servers 在中國

將 Amazon GameLift Servers 用於由 Sinnet 營運之中國（北京）區域中的資源，或由 NWCD 營運之中國（寧夏）區域時，您必須擁有單獨的 AWS（中國）帳戶。請注意，某些功能在中國區域無法使用。如需 Amazon GameLift Servers 在這些區域中使用的詳細資訊，請參閱下列資源：

- [中國的 Amazon Web Services](#)
- [Amazon GameLift Servers](#)（在中國開始使用 Amazon Web Services）

Amazon GameLift Servers 的定價

的定價 Amazon GameLift Servers 取決於您使用的服務類型和功能。如需定價的完整討論，包括範例案例，請參閱 [Amazon GameLift Servers 定價](#)。

您可以使用 免費 AWS 方案來使用 [Amazon GameLift Servers](#) 而不會產生費用。如果您成為 AWS 客戶的時間少於 12 個月，而且您維持在免費方案用量限制內，則可以使用 免費方案優惠。如需詳細資訊，請參閱 [AWS 免費方案](#)。

每個 Amazon GameLift Servers 服務的成本基礎如下：

- Amazon GameLift Servers 受管託管 – 您根據 (1) 託管遊戲工作階段的每小時執行個體用量，以及 (2) 遊戲用戶端與託管遊戲伺服器之間的流量資料傳輸來支付 費用。執行個體使用成本會根據 AWS 區域執行個體類型/大小、作業系統，以及是否為 Spot 或隨需執行個體而有所不同。
- Amazon GameLift Servers FlexMatch 獨立 – 您根據 (1) 您為配對所提出的玩家請求數量，以及 (2) 評估配對請求和提議配對所需的處理時間來支付。
- Amazon GameLift Servers Anywhere 託管 – 您根據 (1) Amazon GameLift Servers 放置在 Anywhere 運算上的遊戲工作階段數目，以及 (2) 伺服器程序連線分鐘數來支付。
- Amazon GameLift Servers FleetIQ – 您需要為遊戲伺服器群組中的每個 EC2 執行個體支付費用。費用衍生自每小時執行個體價格，並且是 EC2 使用成本以外的費用。執行個體使用成本會根據 AWS 區域執行個體類型/大小、作業系統，以及是否為 Spot 或隨需執行個體而有所不同。

主題

- [產生 Amazon GameLift Servers 定價預估](#)
- [管理您的 Amazon GameLift Servers 託管成本](#)

產生 Amazon GameLift Servers 定價預估

使用 AWS 定價計算工具，您可以 [建立 的定價預估 Amazon GameLift Servers](#)。您不需要 AWS 帳戶 或 的深入知識 AWS，即可使用計算器。

AWS 定價計算工具 計算器會引導您完成影響服務成本的決策，讓您了解遊戲專案 Amazon GameLift Servers 可能要支付多少費用。如果您不確定如何計劃使用 Amazon GameLift Servers，請使用預設值來產生預估值。規劃生產用量時，計算器可協助您測試潛在案例，並產生更準確的預估值。

您可以使用 AWS 定價計算工具 來產生下列 Amazon GameLift Servers 託管選項的預估值：

- [估算 Amazon GameLift Servers 受管託管](#)

估算Amazon GameLift Servers受管託管

此選項提供在Amazon GameLift Servers受管伺服器上託管遊戲的成本估算，包括伺服器執行個體用量和資料傳輸的成本。使用 Amazon GameLift Servers受管託管，FlexMatch配對無需額外費用。

如果您要託管或計劃在多個 AWS 區域或多個執行個體類型上託管遊戲伺服器，請為每個區域和執行個體類型建立預估值。

Amazon GameLift Servers 執行個體

本節可協助您預估為玩家託管遊戲工作階段所需的運算資源類型和數量。Amazon GameLift Servers 使用 [Amazon Elastic Compute Cloud \(Amazon EC2\) 執行個體](#) 來管理遊戲伺服器。在 Amazon GameLift Servers，您會部署具有特定執行個體類型和作業系統的執行個體機群。如果您有或計劃有多個機群，請為每個機群建立預估值。

若要開始使用，請開啟 [的設定Amazon GameLift Servers頁面](#) AWS 定價計算工具。新增描述、選擇區域，然後選擇預估Amazon GameLift Servers託管（執行個體 + 資料傳輸輸出）。在Amazon GameLift Servers執行個體下，完成下列欄位：

- 尖峰並行玩家（尖峰 CCU）

這是可同時連線到遊戲伺服器的玩家人數上限。此欄位指出滿足尖峰玩家需求Amazon GameLift Servers所需的託管容量。輸入您預期使用所選 AWS 區域中的執行個體託管的玩家每日尖峰數量。

例如，如果您想要讓 1,000 名玩家一次連接到您的遊戲，請保留預設值 **1000**。

- 每小時平均 CCU 佔每日尖峰 CCU 的百分比

這是 24 小時內每小時並行玩家的平均數量。我們使用這個值來估計需要為玩家Amazon GameLift Servers維護的持續託管容量。如果您不確定要使用的百分比值，請保留預設值**50**百分比。對於玩家需求穩定的遊戲，我們建議輸入 **70%** 的值。

例如，如果您的遊戲每小時平均 CCU 為 6,000，尖峰 CCU 為 10,000，請輸入**60**百分比的值。

- 每個執行個體的遊戲工作階段

這是您的每個遊戲伺服器執行個體可以同時託管的遊戲工作階段數量。可能影響此數量的因素包括遊戲伺服器的資源需求、每個遊戲工作階段中要託管的玩家數量，以及玩家效能期望。如果您知道遊戲的並行遊戲工作階段數量，請輸入該值。或者，保留預設值 **20**。

- 每個遊戲工作階段的玩家數

這是連線到遊戲工作階段的平均玩家人數，如您的遊戲設計所定義。如果您有具有不同玩家數量的遊戲模式，請估計整個遊戲中每個遊戲工作階段的平均玩家數量。預設值為 **8**。

- 執行個體閒置緩衝區 %

這是為了處理玩家需求的突然激增而保留的未使用託管容量百分比。緩衝區大小是機群中執行個體總數的百分比。預設值為 **10** 百分比。

例如，使用 20% 的閒置緩衝區時，支援 100 個作用中執行個體的玩家的機群會維持 20 個閒置執行個體。

- Spot 執行個體 %

Amazon GameLift Servers 機群可以使用隨需執行個體和 Spot 執行個體的組合。雖然隨需執行個體提供更可靠的可用性，但 Spot 執行個體提供具成本效益的替代方案。我們建議您使用組合來最佳化成本節省和可用性。如需 Amazon GameLift Servers 如何使用 Spot 執行個體的資訊，請參閱 [隨需執行個體與 Spot 執行個體](#)。

針對此欄位，輸入要在機群中維護的 Spot 執行個體百分比。我們建議 Spot 執行個體百分比介於 50% 到 85% 之間。預設值為 **50** 百分比。

例如，如果您部署具有 100 個執行個體的機群並指定 **40** 百分比，則 Amazon GameLift Servers 會運作來維護 60 個隨需執行個體和 40 個 Spot 執行個體。

- 執行個體類型

Amazon GameLift Servers 機群可以使用各種 Amazon EC2 執行個體類型，其運算能力、記憶體、儲存體和聯網功能各不相同。當您設定 Amazon GameLift Servers 機群時，請選擇最符合您遊戲需求的執行個體類型。如需使用 選取執行個體類型的詳細資訊 Amazon GameLift Servers，請參閱 [選擇受管機群的運算資源](#)。

如果您知道您正在使用或計劃在 Amazon GameLift Servers 機群中使用的執行個體類型，請選擇該類型。如果您不確定要選擇哪種類型，請考慮選擇 c5.large。這是具有平均大小和功能的高可用性類型。

- 作業系統

此欄位指定遊戲伺服器執行所在的作業系統，無論是 Linux 或 Windows。預設值為 Linux。

資料傳輸 (DTO)

本節可協助您預估遊戲用戶端與遊戲伺服器之間的流量成本。資料傳輸費用僅適用於傳出流量。傳入資料傳輸不收取費用。

在 [的設定Amazon GameLift Servers頁面上](#) AWS 定價計算工具，展開資料傳輸 (DTO)，然後完成下列欄位：

- DTO 預估類型

您可以選擇以下列兩種方式之一來估計 DTO，具體取決於您追蹤遊戲資料傳輸的方式。

- 每月（以 GB 為單位）– 如果您追蹤遊戲伺服器的每月流量，請選擇此類型。
- 每個玩家 – 如果您追蹤玩家的資料傳輸，請選擇此類型。這是預設類型。

在下列欄位中，您可以根據您在上一節中計算的玩家時數來估計每個玩家的 DTO。

- 每月 DTO（以 GB 為單位）

如果您選擇每月（以 GB 為單位）DTO 估算類型，則輸入每個區域每個執行個體的每月 DTO 用量估計值，以 GB 為單位。

- 每個玩家的 DTO

如果您選擇每個玩家的 DTO 預估類型，則請以 KB/秒為單位，輸入每個玩家的遊戲預估 DTO 用量。預設值為 4。

當您完成Amazon GameLift Servers定價預估的設定後，請選擇新增至我的預估。如需在 中建立和管理預估的詳細資訊 AWS 定價計算工具，請參閱AWS 定價計算工具 《使用者指南》中的[建立預估、設定服務，以及新增更多服務](#)。

管理您的Amazon GameLift Servers託管成本

您的 AWS 帳單會反映您的遊戲託管成本。您可以在帳單主控台上檢視當月的預估費用，以及前幾個月的最終費用，網址為 <https://console.aws.amazon.com/costmanagement/>。如需協助您管理 AWS 成本的工具和資源詳細資訊，請參閱[AWS Billing 《使用者指南》](#)。本指南可協助您檢閱資源消耗、建立未來用量，以及判斷您的擴展需求。

請考慮這些秘訣，以協助您管理 Amazon GameLift Servers 服務的成本。

建立帳單提醒以監控用量

設定 AWS 免費方案用量提醒，以在用量接近或超過 Amazon GameLift Servers 和其他的免費方案限制時通知您 AWS 服務。您可以設定提醒，根據您的用量層級採取動作。例如，當達到免費方案限制時，您可以自動將預算設定為零。

您也可以設定 Amazon CloudWatch 帳單提醒，以在用量達到自訂閾值時收到通知。

如需詳細資訊，請參閱 AWS Billing 《使用者指南》中的這些主題：

- [追蹤您的 AWS 免費方案用量](#)
- [帳單提醒偏好設定](#)

追蹤每個 Amazon GameLift Servers 機群的成本

使用 AWS 成本分配標籤，根據 Amazon EC2 Amazon GameLift Servers 機群和其他資源來組織和追蹤您的遊戲託管成本。透過個別或群組標記機群，您可以建立成本分配報告，根據指派的標籤來分類成本。您可以使用此類型的報告來識別機群如何為您的託管成本做出貢獻。您也可以使用標籤來篩選 AWS Cost Explorer 中的檢視。

如需詳細資訊，請參閱以下主題：

- [使用 AWS 成本分配標籤](#)、AWS Billing 使用者指南
- [使用 AWS Cost Explorer 分析您的成本](#)，AWS Cost Management 使用者指南

將受管機群容量設定為零

即使未用於託管遊戲工作階段，受管機群仍可能會持續產生成本。為了避免產生不必要的費用，請在不使用時將機群縮減為零。如果您使用自動擴展，請暫停此活動並手動設定機群容量。

Amazon GameLift Servers 入門

利用這些入門資源，進一步了解 Amazon GameLift Servers 服務，以及如何開始為工作階段型多玩家遊戲開發自訂託管解決方案。

開始之前

- 建立 AWS 帳戶（或指定現有）以搭配使用 Amazon GameLift Servers。
- 設定具有 Amazon GameLift Servers 和相關 AWS 服務許可的使用者。
- 選取要 AWS 區域使用的。針對開發，請選擇靠近您位置的區域。您可以隨時變更區域。

[設定 AWS 帳戶](#)

快速加入選項

試用這些快速入門工具，透過簡化的開發快速啟動和執行基本託管解決方案。這些工具非常適合用於概念驗證和原型設計，或使用它們來建置測試環境，以快速反覆開發遊戲。使用這些工具部署遊戲伺服器進行託管後，您可以使用 Amazon GameLift Servers 主控台和 API 工具來監控機群效能、管理遊戲工作階段和分析指標。

- [適用於的遊戲伺服器包裝函式 Amazon GameLift Servers](#) – 此工具是使用讓遊戲伺服器託管和執行遊戲工作階段最快速且最簡單的方式 Amazon GameLift Servers，不需要變更遊戲程式碼。遊戲伺服器包裝函式提供基本遊戲工作階段管理功能和簡化的遊戲伺服器部署。它非常適合 Amazon GameLift Servers 使用您自己的遊戲專案或範例專案進行實作評估。當您準備好建置自訂遊戲託管解決方案時，請切換到與適用於的伺服器 SDK 完全整合的其中一個自訂開發選項 Amazon GameLift Servers。如果您的遊戲不需要自訂託管解決方案，您可以繼續使用遊戲伺服器包裝函式在生產環境中部署和託管遊戲伺服器。
- [Amazon GameLift Servers Unreal Engine 或 Unity 的外掛程式](#) – 外掛程式為您提供 GUI 工作流程和範例資產，以引導您完成初始步驟，並使用基本託管解決方案部署遊戲伺服器。使用外掛程式設定自我管理的 Anywhere 機群託管，或部署雲端型、受管的 EC2 機群或容器機群。當您準備好開發自訂託管解決方案時，您可以建置在外掛程式建置的解決方案上。
- 適用於 [Amazon GameLift Servers 受管容器的入門套件](#) – 此套件可簡化整合遊戲伺服器、準備遊戲伺服器容器映像，以及部署容器機群以進行託管的任務。為了整合，套件會將必要的遊戲工作階段管理功能新增至您的遊戲伺服器。套件使用預先設定的範本，為遊戲伺服器建置容器機群和自動化部署

管道。當您準備好新增完整的遊戲工作階段管理功能時，請遵循其中一個自訂開發藍圖來整合適用於的伺服器 SDK Amazon GameLift Servers。

自訂開發選項

遵循其中一個開發藍圖，開始為您的遊戲建置功能完整的自訂託管解決方案。藍圖提供有關如何在託管解決方案中建立、測試和自訂每個元件的詳細指導。

- [使用 Amazon GameLift Servers 受管 EC2 託管的開發藍圖](#)
- [使用 Amazon GameLift Servers 受管容器託管的開發藍圖](#)
- [使用 Amazon GameLift Servers Anywhere 託管的開發藍圖](#)
- [使用 混合託管的開發藍圖 Amazon GameLift Servers](#)

Amazon GameLift Servers 範例

如果您考慮使用 Amazon GameLift Servers 來管理您的自訂遊戲伺服器，或有興趣利用 Amazon GameLift Servers Realtime，建議您先嘗試下列範例，再將服務用於您自己的遊戲。自訂遊戲伺服器範例為您提供在 Amazon GameLift Servers 主控台中託管遊戲的體驗。此 Amazon GameLift Servers Realtime 範例說明如何準備使用 Realtime 伺服器託管的遊戲。

自訂遊戲伺服器範例

此範例示範將範例遊戲伺服器部署到 Amazon GameLift Servers 受管 EC2 機群以進行託管的程序。使用範例遊戲用戶端連線到即時遊戲工作階段。您可以體驗如何使用 Amazon GameLift Servers .tools，包括 主控台和 AWS CLI，來監控機群的託管效能和用量。

此範例會逐步引導您完成下列步驟：

- 上傳範例遊戲伺服器建置。
- 建立機群以執行遊戲伺服器建置。
- 取得範例遊戲用戶端，並使用它連線到遊戲伺服器並加入遊戲工作階段。
- 檢閱機群和遊戲工作階段指標。

啟動多個遊戲用戶端並玩遊戲以產生託管資料。使用 Amazon GameLift Servers 主控台檢視託管資源、追蹤指標，以及探索擴展機群託管容量的選項。

若要開始使用，請登入 [Amazon GameLift Servers主控台](#)。在左側導覽中，前往資源，嘗試範例遊戲。

Amazon GameLift ServersRealtime 範例

此範例是完整的教學課程，引導您如何搭配 部署範例多玩家遊戲 Mega Frog RaceAmazon GameLift ServersRealtime。本教學課程涵蓋如何整合您的遊戲用戶端與 Realtime SDK，以及將完整的託管解決方案與受管 EC2 機群上的Realtime伺服器部署。

如需實作教學課程，請參閱[適用於遊戲的 部落格上的使用幾行 JavaScript 為多玩家行動遊戲建立伺服器](#)。AWS 如需 Mega Frog Race 的原始碼，請參閱 [GitHub 儲存庫](#)。

原始程式碼包含下列部分：

- 遊戲用戶端 – C++ 遊戲用戶端的原始程式碼，在 Unity 中建立。遊戲用戶端會取得遊戲工作階段連線資訊、連線至伺服器，以及與其他玩家交換更新。
- 後端服務 – AWS Lambda 函數的原始程式碼，用於管理對 服務 API 的直接呼叫Amazon GameLift Servers。
- Realtime 指令碼 – 來源指令碼檔案，可設定遊戲的Realtime伺服器機群。此指令碼包含每個 Realtime伺服器與 通訊Amazon GameLift Servers和託管遊戲工作階段所需的最低組態。

設定託管的範例遊戲之後，請使用它做為試驗其他Amazon GameLift Servers功能的起點，例如 FlexMatch。

設定 AWS 帳戶

若要開始使用 Amazon GameLift Servers，請建立並設定您的 AWS 帳戶。建立 無需付費 AWS 帳戶。本節會逐步引導您建立 帳戶、設定 使用者，以及設定 許可。

主題

- [註冊 AWS 帳戶](#)
- [建立具有管理存取權的使用者](#)
- [設定 的使用者許可 Amazon GameLift Servers](#)
- [為使用者設定程式設計存取](#)
- [為您的遊戲設定程式設計存取](#)
- [的 IAM 許可範例 Amazon GameLift Servers](#)
- [設定的 IAM 服務角色 Amazon GameLift Servers](#)

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

1. 開啟 <https://portal.aws.amazon.com/billing/signup>。
2. 請遵循線上指示進行。

部分註冊程序需接收來電或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行[需要根使用者存取權的任務](#)。

AWS 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶您的電子郵件地址，以帳戶擁有者[AWS Management Console](#)身分登入。在下一頁中，輸入您的密碼。

如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的[以根使用者身分登入](#)。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

如需說明，請參閱《IAM 使用者指南》中的[為您的 AWS 帳戶根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

1. 啟用 IAM Identity Center。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[啟用 AWS IAM Identity Center](#)。

2. 在 IAM Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

以具有管理存取權的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

如需使用 IAM Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》中的[登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[建立許可集](#)。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[新增群組](#)。

設定 的使用者許可 Amazon GameLift Servers

視需要為您的 Amazon GameLift Servers 資源建立其他使用者或將存取許可擴展至現有使用者。最佳實務 ([IAM 中的安全最佳實務](#)) 為所有使用者套用最低權限許可。如需許可語法的指引，請參閱 [的 IAM 許可範例 Amazon GameLift Servers](#)。

請依照您管理 AWS 帳戶中使用者的方式，使用下列指示來設定使用者許可。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 中的使用者和群組 AWS IAM Identity Center：

建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 [建立權限合集](#) 說明進行操作。

- 透過身分提供者在 IAM 中管理的使用者：

建立聯合身分的角色。遵循「IAM 使用者指南」的[為第三方身分提供者 \(聯合\) 建立角色](#)中的指示。

- IAM 使用者：

- 建立您的使用者可擔任的角色。請按照「IAM 使用者指南」的[為 IAM 使用者建立角色](#)中的指示。
- (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的[新增許可到使用者 \(主控台\)](#) 中的指示。

使用 IAM 使用者時，最佳實務是一律將許可連接到角色或使用者群組，而非個別使用者。

為使用者設定程式設計存取

如果使用者想要與 AWS 外部互動，則需要程式設計存取 AWS Management Console。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	根據
人力資源身分 (IAM Identity Center 中管理的 使用者)	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的設定 AWS CLI 要使用 AWS IAM Identity Center的。 • AWS SDKs、工具和 AWS APIs，請參閱 AWS SDKs 和工具參考指南中的IAM Identity Center 身分驗證。
IAM	使用暫時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	遵循《IAM 使用者指南》中 將臨時登入資料與 AWS 資源搭配使用 的指示。
IAM	(不建議使用) 使用長期登入資料來簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱 AWS Command Line Interface

哪個使用者需要程式設計存取權？	到	根據
		《使用者指南》中的使用 IAM 使用者憑證進行身分驗證。 • AWS SDKs和工具，請參閱 AWS SDKs和工具參考指南中的使用長期憑證進行身分驗證。 • 對於 AWS APIs，請參閱《IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

如果您使用存取金鑰，請參閱[管理 AWS 存取金鑰的最佳實務](#)。

為您的遊戲設定程式設計存取

大多數遊戲使用後端服務來Amazon GameLift Servers使用 AWS SDKs與 通訊。使用後端服務（遊戲用戶端的動作）來請求遊戲工作階段、將玩家放入遊戲和其他任務。這些服務需要程式設計存取和安全登入資料，才能驗證對 服務 API 的呼叫Amazon GameLift Servers。

對於 Amazon GameLift Servers，您可以透過在 AWS Identity and Access Management (IAM) 中建立玩家使用者來管理此存取權。透過下列其中一個選項管理玩家使用者許可：

- 建立具有玩家使用者許可的 IAM 角色，並允許玩家使用者在需要時擔任該角色。在向 提出請求之前，後端服務必須包含擔任此角色的程式碼Amazon GameLift Servers。根據安全最佳實務，角色提供有限的暫時存取。您可以針對在 AWS 資源 ([IAM 角色](#)) 或 AWS ([IAM Roles Anywhere](#)) 外部執行的工作負載使用 角色。
- 建立具有玩家使用者許可的 IAM 使用者群組，並將您的玩家使用者新增至群組。此選項提供玩家使用者長期登入資料，後端服務在與 通訊時必須存放和使用Amazon GameLift Servers。

如需許可政策語法，請參閱 [玩家使用者許可範例](#)。

如需管理工作負載使用許可的詳細資訊，請參閱 [IAM 身分：IAM 中的臨時憑證](#)。

的 IAM 許可範例 Amazon GameLift Servers

使用這些範例中的語法，為需要存取 Amazon GameLift Servers 資源的使用者設定 AWS Identity and Access Management (IAM) 許可。如需管理使用者許可的詳細資訊，請參閱 [設定的使用者許可 Amazon GameLift Servers](#)。管理 IAM Identity Center 外部使用者的許可時，最佳實務一律將許可連接至 IAM 角色或使用者群組，而非個別使用者。

如果您使用 Amazon GameLift ServersFleetIQ 做為獨立解決方案，請參閱 [設定 AWS 帳戶的 Amazon GameLift ServersFleetIQ](#)。

管理許可範例

這些範例提供託管管理員或開發人員目標存取權，以管理 Amazon GameLift Servers 遊戲託管資源。

Example Amazon GameLift Servers 完整存取資源許可的語法

下列範例會將完整存取權延伸到所有 Amazon GameLift Servers 資源。

JSON

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

Example 資源 Amazon GameLift Servers 許可的語法，支援預設未啟用的區域

下列範例會將存取權延伸至預設未啟用的所有 Amazon GameLift Servers 資源和 AWS 區域。如需未預設啟用的區域以及如何啟用它們的詳細資訊，請參閱《》中的 [管理 AWS 區域](#) AWS 一般參考。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
```



```
"Action": [
    "ec2:DescribeRegions",
    "gamelift:*"
],
"Resource": "*"
}
```

Example 資源Amazon GameLift Servers在 Amazon ECR 中存取容器映像的語法

下列範例會延伸存取Amazon GameLift Servers使用者在使用受管容器機群時所需的 Amazon Elastic Container Registry (Amazon ECR) 動作。

JSON

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

Example Amazon GameLift Servers 資源和**PassRole**許可的語法

下列範例會延伸對所有 Amazon GameLift Servers 資源的存取，並允許使用者將 IAM 服務角色傳遞給 Amazon GameLift Servers。服務角色可讓您Amazon GameLift Servers有限度地代表您存取其他資源和服務，如中所述[設定的 IAM 服務角色 Amazon GameLift Servers](#)。例如，回應CreateBuild請求時，Amazon GameLift Servers需要存取 Amazon S3 儲存貯體中的建置檔案。如需 PassRole動作的詳細資訊，請參閱 [《IAM 使用者指南》中的 IAM：將 IAM 角色傳遞至特定 AWS 服務](#)。

JSON

```
{
```

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

玩家使用者許可範例

這些範例允許後端服務或其他實體對 API 進行 Amazon GameLift Servers API 呼叫。它們涵蓋了管理遊戲工作階段、玩家工作階段和配對的常見案例。如需詳細資訊，請參閱[為您的遊戲設定程式設計存取](#)。

Example 遊戲工作階段置放許可的語法

下列範例會將存取權延伸到使用遊戲工作階段置放佇列來建立遊戲工作階段和管理玩家工作階段的 Amazon GameLift Servers APIs。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "PlayerPermissionsForGameSessionPlacements",
    "Effect": "Allow",
    "Action": [
      "gamelift:StartGameSessionPlacement",
      "gamelift:DescribeGameSessionPlacement",
```

```
    "gamelift:StopGameSessionPlacement",
    "gamelift:CreatePlayerSession",
    "gamelift:CreatePlayerSessions",
    "gamelift:DescribeGameSessions"
  ],
  "Resource": "*"
}
```

Example 配對許可的語法

下列範例會將存取權延伸到管理FlexMatch配對活動的 Amazon GameLift Servers APIs。FlexMatch會比對新遊戲工作階段或現有遊戲工作階段的玩家，並為上託管的遊戲啟動遊戲工作階段放置Amazon GameLift Servers。如需的詳細資訊FlexMatch，請參閱[什麼是Amazon GameLift ServersFlexMatch？](#)

JSON

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

Example 手動遊戲工作階段置放許可的語法

下列範例延伸對 API 的存取，這些 Amazon GameLift Servers API 會在指定的機群上手動建立遊戲工作階段和玩家工作階段。APIs 此案例支援不使用置放佇列的遊戲，例如透過從可用遊戲工作階段（「list-and-pick」方法）清單中選擇讓玩家加入的遊戲。

JSON

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

設定 的 IAM 服務角色 Amazon GameLift Servers

某些Amazon GameLift Servers功能需要您限制存取您擁有的其他 AWS 資源。您可以建立 AWS Identity and Access Management (IAM) 角色來執行此操作。[IAM 角色](#)是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。IAM 角色類似於 IAM 使用者，因為它是具有許可政策的 AWS 身分，可決定身分可以和不可以執行的操作 AWS。但是，角色的目的是讓需要它的任何人可代入，而不是單獨地與某個人員關聯。此外，角色沒有與之關聯的標準長期憑證，例如密碼或存取金鑰。反之，當您擔任角色時，其會為您的角色工作階段提供臨時安全性憑證。

本主題說明如何建立可與Amazon GameLift Servers受管機群搭配使用的角色。如果您使用 Amazon GameLift ServersFleetIQ 來最佳化 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體上的遊戲託管，請參閱[設定 AWS 帳戶 的 Amazon GameLift ServersFleetIQ](#)。

在下列程序中，建立具有自訂許可政策和信任政策的角色，Amazon GameLift Servers允許 擔任該角色。

為Amazon GameLift Servers受管 EC2 機群建立 IAM 服務角色

步驟 1：建立許可政策。

使用本頁面上的指示和範例，為您正在使用的Amazon GameLift Servers機群類型建立自訂許可政策。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。

3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入或貼上 JSON 政策文件。如需有關 IAM 政策語言的詳細資訊，請參閱 [IAM JSON 政策參考](#)。
6. 解決[政策驗證](#)期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。

Note

您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的[調整政策結構](#)。

7. (選用) 當您在 中建立或編輯政策時 AWS Management Console，您可以產生可在 範本中使用的 JSON 或 YAML 政策 AWS CloudFormation 範本。

若要執行此動作，請在政策編輯器中選擇動作，然後選擇產生 CloudFormation 範本。若要進一步了解 AWS CloudFormation，請參閱AWS CloudFormation 《使用者指南》中的[AWS Identity and Access Management 資源類型參考](#)。

8. 將許可新增至政策後，請選擇下一步。
9. 在檢視與建立頁面上，為您在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
10. (選用) 藉由連接標籤作為鍵值組，將中繼資料新增至政策。如需在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的[AWS Identity and Access Management 資源的標籤](#)。
11. 選擇 Create policy (建立政策) 儲存您的新政策。

步驟 2：建立Amazon GameLift Servers可擔任的角色。

若要建立一個 IAM 角色

1. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
2. 在選取信任實體頁面上，選擇自訂信任政策選項。此選項會開啟自訂信任政策編輯器。
3. 將預設 JSON 語法取代為以下內容，然後選擇下一步以繼續。

JSON

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

4. 在新增許可頁面上，尋找並選取您在步驟 1 中建立的許可政策。選擇 Next (下一步) 繼續。
5. 在名稱、檢閱和建立頁面上，輸入您要建立之角色的角色名稱和描述（選用）。檢閱信任實體和新增的許可。
6. 選擇建立角色以儲存您的新角色。

為Amazon GameLift Servers受管容器建立 IAM 角色

如果您使用的是Amazon GameLift Servers受管容器，則需要建立 IAM 服務角色，以便與容器機群搭配使用。此角色會授予Amazon GameLift Servers管理容器機群資源所需的有限許可，並代表您採取動作。

為容器機群建立 IAM 角色

1. 登入 AWS Management Console 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 在選取信任的實體頁面上，選擇AWS 服務，然後選取使用案例「GameLift」。選擇下一步

4. 在新增許可上，選擇 受管政策 GameLiftContainerFleetPolicy。選擇下一步。如需此政策的詳細資訊[AWS 的 受管政策 Amazon GameLift Servers](#)，請參閱。
5. 在名稱、檢閱和建立上，輸入角色名稱，然後選擇建立角色以儲存新角色。

許可政策語法

- Amazon GameLift Servers擔任服務角色的許可

JSON

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

- 存取預設未啟用之 AWS 區域的許可

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "gamelift.amazonaws.com",
          "gamelift.ap-east-1.amazonaws.com",
          "gamelift.me-south-1.amazonaws.com",
          "gamelift.af-south-1.amazonaws.com",
          "gamelift.eu-south-1.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```



```
        "Action": "sts:AssumeRole"
    }
  ]
}
```

取得Amazon GameLift Servers開發工具

Amazon GameLift Servers 提供一組SDKs和其他工具，協助您為遊戲建置遊戲託管解決方案。SDKs 將功能新增至遊戲伺服器、遊戲用戶端和後端服務，讓它們能夠與服務互動Amazon GameLift Servers。如需 Amazon GameLift Servers SDK 版本和相容性的最新資訊，請參閱 [Amazon GameLift Servers 版本備註](#)。

對於遊戲伺服器

整合並建置您的 64 位元遊戲伺服器與適用於 的伺服器 SDKAmazon GameLift Servers。遊戲伺服器使用伺服器 SDK 與 Amazon GameLift Servers服務通訊以進行遊戲工作階段管理，包括啟動、更新和停止遊戲工作階段。如需將伺服器 SDK 整合到您的遊戲專案的說明，請參閱 [準備 的遊戲 Amazon GameLift Servers](#)。

開發支援

- 開發作業系統
 - Windows
 - Linux
- 程式設計語言

[取得 Amazon GameLift Servers SDK](#)。如需版本特定資訊和安裝說明，請參閱每個套件中包含的讀我檔案。

- [C++ 伺服器開發套件](#)
 - [伺服器開發套件參考](#)
 - [如何整合](#)
- [C# 伺服器 SDK](#)（支援 .NET 4、.NET 6、.NET 8 因版本而異，請參閱 [軟體開發套件版本](#)）
 - [伺服器開發套件參考](#)
 - [如何整合](#)
- [Go 伺服器開發套件](#)
 - [伺服器開發套件參考](#)

- [如何整合](#)
- 遊戲引擎支援

的完整功能外掛程式Amazon GameLift Servers包括 UI 工作流程和範例資產，以及 AWS SDK 和伺服器 SDK 的內建版本。工作流程會引導您如何設定和部署遊戲伺服器，以託管受管 EC2 機群、受管容器機群或自我管理的 Anywhere 機群。如果您不需要引導式工作流程，您可以從相同的 GitHub 儲存庫取得遊戲引擎的伺服器開發套件。

如果您使用的是外掛程式不支援的另一個遊戲引擎或開發環境，請取得程式設計語言的伺服器 SDK，並將其新增至您的遊戲專案。

如需版本特定資訊和安裝說明，請參閱每個套件中包含的讀我檔案。

- [Unreal Engine 的外掛程式](#) – 適用於 Unreal Engine 5.0、5.1、5.2、5.3、5.4、5.5 和 5.6 版。檢查版本特定的讀我檔案是否有 Unreal 支援。
 - [Unreal Engine 的外掛程式指南](#)
 - [適用於 -- 動作的 C++ \(非真實\) 伺服器 SDK Amazon GameLift Servers 5.x](#)
- [Unity 的外掛程式](#) – 專為搭配 Unity Editor 6.0、2022.3 或 2021.3 的 LTS 版本而打造。它支援 Unity 的 .NET Framework 和 .NET Standard 設定檔，以及 .NET Standard 2.1 和 .NET 4.x。檢查 Unity 支援的特定版本讀取。
 - [Unity 的外掛程式指南](#)
 - [C# 伺服器 SDK 參考](#)
- [適用於 Unreal 的伺服器 SDK](#)
 - [伺服器開發套件參考](#)
 - [Amazon GameLift Servers整合至 Unreal Engine 專案](#)
- [適用於 Unity 的伺服器 SDK](#)
 - [伺服器開發套件參考](#)
 - [Amazon GameLift Servers整合至 Unity 專案](#)

執行期支援

對於受管託管解決方案，請建置您的遊戲伺服器以在下列其中一個 Amazon 機器映像 (AMIs) 上執行。如需詳細資訊[Amazon GameLift Servers AMI 版本](#)，Amazon GameLift Servers請參閱。

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

- [Amazon Linux 2](#)

 Note

Amazon Linux 2 (AL2) 將於 2025 年 6 月 30 日終止支援。如需詳細資訊，請參閱 [Amazon Linux 2 FAQs](#)。對於託管於 AL2 並使用 Amazon GameLift Servers 伺服器 SDK 4.x 的遊戲伺服器，請先將遊戲伺服器組建更新為伺服器 SDK 5.x，然後部署至 AL2023 執行個體。請參閱 [遷移至適用於的伺服器 SDK 5.x Amazon GameLift Servers](#)。

其他工具

[的遊戲伺服器包裝函式 Amazon GameLift Servers](#)

此工具可協助您部署遊戲伺服器，以託管一組基本遊戲工作階段管理功能。使用此工具，您不需要變更遊戲程式碼或整合適用於的伺服器 SDK Amazon GameLift Servers。使用遊戲伺服器包裝函式來封裝遊戲伺服器，並使用三個託管解決方案 (Anywhere、受管 EC2 或受管容器) 中的任何一個來部署遊戲 Amazon GameLift Servers 託管。此工具最適合使用您自己的遊戲或範例遊戲進行早期評估或原型設計，因為此工具不支援遊戲伺服器自訂。如果您的遊戲不需要自訂功能，您可以使用遊戲伺服器包裝函式部署遊戲伺服器以進行生產託管。

[Amazon GameLift Servers 工具組](#)

Toolkit Amazon GameLift Servers 是指令碼和其他工具的集合，我們開發這些工具來協助開發人員處理常見的案例和問題。工具組資料包括指令碼、範本程式碼和讀我檔案。

- [容器入門套件](#) – 使用此工具可簡化設定遊戲伺服器組建以託管 Amazon GameLift Servers 受管容器的任務。套件將基本遊戲工作階段管理功能整合到遊戲伺服器，並使用預先設定的範本來建立容器機群，並為遊戲伺服器建置設定自動化部署管道。部署之後，您可以使用 Amazon GameLift Servers 主控台和 API 工具來監控機群效能、管理遊戲工作階段和分析指標。套件與 AWS CodeBuild 整合，用於建置自動化、Amazon Simple Storage Service 用於儲存，以及 AWS CloudFormation 用於基礎設施部署。
- [快速建置更新工具](#) – 使用此工具來修改已部署到受管 EC2 機群的遊戲伺服器建置。此工具旨在協助您快速交換遊戲建置檔案，而不必在每次變更時設定和建立新的 EC2 機群。您可以更新機群中的個別執行個體或所有執行個體。選項可讓您取代特定建置檔案或整個建置，並讓您在更新後管理如何重新啟動遊戲伺服器。

對於遊戲用戶端服務

使用 AWS SDK 為您的遊戲用戶端建立 64 位元後端服務，其中包含的服務 API Amazon GameLift Servers。您的遊戲的後端服務會處理與 Amazon GameLift Servers 服務的用戶端互動，包括開始新的遊戲工作階段，以及將玩家加入遊戲。

[取得 AWS SDK](#)

如需搭配使用 AWS SDK 的詳細資訊 Amazon GameLift Servers，請參閱下列資源：

- [Amazon GameLift Servers API 參考](#)
- [用戶端服務整合](#)
- [設計用戶端後端服務](#)

用於 Amazon GameLift Servers 資源管理

使用下列工具來建立、更新和監控您的 Amazon GameLift Servers 受管託管資源。

- [AWS Management Console](#) – AWS 主控台是一種以 Web 為基礎的應用程式，可讓您集中存取所有個別 AWS 服務主控台，包括 Amazon GameLift Servers。使用主控台建立或登入 AWS 帳戶，並開啟 Amazon GameLift Servers 主控台以使用您的遊戲託管資源。您可以設定和部署託管機群和其他資源、檢視用量和效能指標、在儀表板中追蹤資源，以及許多其他任務。[前往 Amazon GameLift Servers 主控台。](#)
- [的服務 API Amazon GameLift Servers](#) – 此 API 可讓您以程式設計方式存取所有 Amazon GameLift Servers 資源。這是 SDK 的一部分，您可以下載此 AWS 開發套件以搭配最熱門的程式設計語言使用。[取得 AWS SDK。](#)
- [AWS 命令列界面 \(CLI\)](#) – AWS CLI 可讓您使用命令列 Shell 與 AWS 服務互動。這些工具可讓您直接存取 AWS 服務的公有 APIs，以及可供服務使用的自訂命令。[取得 AWS CLI。](#)
- [AWS CloudFormation](#) for Amazon GameLift Servers – AWS CloudFormation 服務可協助您建立模型和設定 AWS 資源，以簡化基礎設施部署和管理。建立 AWS CloudFormation 範本來描述託管解決方案 Amazon GameLift Servers 的資源，然後使用範本來建置其他資源或更新組態。檢視 [Amazon GameLift Servers 資源類型參考](#)。

對於 Amazon GameLift ServersRealtime

設定和部署Realtime伺服器以託管您的多玩家遊戲。若要允許遊戲用戶端連線到Realtime伺服器，請使用 Amazon GameLift ServersRealtime用戶端 SDK。若要開始使用，[請下載Realtime用戶端 SDK](#)。如需組態資訊，請參閱 [整合的遊戲用戶端 Amazon GameLift ServersRealtime](#)。

開發套件支援

Realtime 用戶端開發套件包含適用於以下語言的來源碼：

- C# (.NET)

開發環境

針對下列支援的開發作業系統和遊戲引擎，視需要從來源建置開發套件：

- 作業系統 – Windows、Linux、Android、iOS
- 遊戲引擎 – Unity，支援 C# 程式庫的引擎

遊戲伺服器作業系統

您可以將Realtime伺服器部署到在下列平台上執行的託管資源：

- [Amazon Linux](#)
- [Amazon Linux 2](#)

Note

AL2 即將結束支援。如需詳細資訊，請參閱 [Amazon Linux 2 FAQs](#)。

教學課程：使用Amazon GameLift Servers包裝函式快速入門

歡迎使用 的入門教學課程Amazon GameLift Servers。在本教學課程中，您會快速部署遊戲伺服器，以託管在雲端型運算資源的機群上。使用此教學課程略過將適用於的伺服器 SDK 整合Amazon GameLift Servers到您的遊戲程式碼的工作，並改為使用最少的功能部署您的遊戲，以與服務通訊Amazon GameLift Servers並執行遊戲工作階段。您將設定基本託管解決方案，並使用它來體驗各種功能，例如自動擴展和配對。這也是在即時示範或測試中託管遊戲原型的絕佳方式。

此加入方法的主要優點：

- 快速部署遊戲伺服器以快速託管。
- 零遊戲程式碼變更且不需要修改。
- 將此方法與任何遊戲可執行檔搭配使用，無論遊戲引擎為何。
- 探索所有Amazon GameLift Servers管理工具，包括監控遊戲工作階段活動和託管運作狀態。

Note

包裝函式適用於評估和基本生產用途。進階功能，例如詳細的玩家工作階段管理，需要完整的伺服器 SDK 整合。

先決條件

開始之前，請確定您已：

- AWS 帳戶 具有適當許可的
- AWS CLI 已安裝
- Go 1.18+
- 多玩家遊戲伺服器可執行檔
- Make (Linux/Mac)
- 與作用中帳戶一起安裝的 Git

概觀

在本教學課程中，您將：

1. 取得並建置包裝函式
2. 準備遊戲組建
3. 設定包裝函式
4. 上傳遊戲伺服器組建
5. 建立受管 EC2 機群
6. 建立並連線至遊戲工作階段

7. 監控和管理遊戲伺服器
8. 擴展您的遊戲伺服器

步驟 1：取得並建置遊戲伺服器包裝函式

使用下列命令取得遊戲伺服器包裝函式來源並建置包裝函式。這些命令使用 SSH，但您也可以直接前往 Github 儲存庫。

Windows

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

Mac 和 Linux

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

在成功的建置上，「輸出」目錄會新增至 `amazon-gamelift-servers-game-server-wrapper`。在此目錄中有三個資料夾，每個支援的託管機群選項各一個，其中包含一組建置成品。在本教學課程中，您將部署至受管 EC2 機群，因此您將使用資料夾 `gamelift-servers-managed-ec2`。

步驟 2：準備您的遊戲伺服器建置

在此步驟中，您會準備遊戲伺服器建置檔案以上傳至 Amazon GameLift Servers。

建立遊戲目錄

現在，在本機電腦上準備遊戲目錄。此目錄必須包含使用執行遊戲伺服器所需的所有檔案 Amazon GameLift Servers。這包括遊戲伺服器包裝函式、遊戲伺服器建置，以及可讓包裝函式與遊戲伺服器搭配使用的 `config.yaml`。

使用下列步驟：

1. 受管 EC2 機群。在遊戲伺服器包裝函式輸出資料夾中，找到要部署到受管 EC2 機群的建置成品。組建將寫入輸出目錄，如下所示：out\linux\amd64\gamelift-servers-managed-ec2。
2. 複製遊戲伺服器可執行檔和執行到 gamelift-servers-managed-ec2 資料夾所需的所有相關檔案。您可以視需要使用巢狀目錄。

範例目錄結構如下所示：

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

步驟 3：設定機群的包裝函式

Amazon GameLift Servers 會管理機群運算執行個體的生命週期，在已安裝伺服器建置的情況下啟動新執行個體，並視需要回收執行個體。服務會管理在每個執行個體上執行的遊戲伺服器程序生命週期。Managed EC2 機群可以在多個位置擁有執行個體，無論玩家身在何處都能提供支援。

編輯 config.yaml 檔案以設定記錄、連接埠設定和伺服器初始化的包裝函式。

1. 設定記錄設定。遊戲伺服器包裝函式會為每個遊戲伺服器程序產生日誌訊息。根據預設，日誌層級會設定為偵錯，以取得最大詳細程度。這在設定和故障診斷期間非常有用，並決定日誌訊息的詳細程度 - 在這種情況下，是最詳細的。選項包括偵錯、資訊、警告和錯誤（最低詳細程度）。
2. 指定遊戲伺服器日誌目錄的路徑。遊戲伺服器日誌的預設路徑為 ./game-server-logs。此目錄包含遊戲伺服器產生的所有日誌，且每個執行個體都有這些日誌。日誌會自動上傳至從事件索引標籤存取日誌Amazon GameLift Servers的位置。如需詳細資訊，請參閱故障診斷一節。
3. 定義網路連接埠組態。將遊戲連接埠設定為您偏好的連接埠。在本教學課程中，請僅指定一個連接埠，因為您將建立一個機群，每個執行個體只會執行一個並行遊戲伺服器程序。如果您決定一次執行多個程序，則需要為每個並行程序設定足夠的連接埠。預設值為 37016，如組態檔案中所示，但一般而言，對於使用 Linux 組建的機群，使用連接埠 22 和 1026-60000。對於使用 Windows 組建的機群，請使用連接埠 1026-60000。

4. 設定遊戲伺服器可執行檔的路徑。對於使用其實際名稱和位置 `./MyGame/my-server-executable.exe` 自訂遊戲伺服器可執行檔的路徑。這是啟動遊戲伺服器的進入點。
5. 設定遊戲伺服器引數。至少，指定 `-port` 引數，並使用您之前定義的相同遊戲連接埠值。"pos" 值 0 表示這是第一個引數。視需要新增其他引數。這些引數會在啟動時傳遞到您的遊戲伺服器，讓您設定其執行時間行為。
 - a. 引數：`"--port"`
 - b. 值：`"{{.port number here}}"`
 - c. 位置：0 (清單中的第一個引數)

範例組態：

```
log-config:
  wrapper-log-level: debug
game-server-logs-dir: ./game-server-logs
ports:
  gamePort: 37016
game-server-details:
  executable-file-path: ./MyGame/my-server-executable
game-server-args:
  - arg: "--port"
    val: "{{.gameport}}"
    pos: 0
```

步驟 4：上傳遊戲伺服器組建

您現在已完成遊戲伺服器組建的所有必要元素（遊戲伺服器包裝函式、`config.yaml` 和遊戲伺服器檔案），並準備好將遊戲組建上傳至 Amazon GameLift Servers 進行託管。上傳遊戲組建的最快速方法是使用 AWS CLI 命令 `upload-build`，如下列範例所示。

使用 Windows 上傳遊戲組建：

```
aws gamelift upload-build \
  --name gamelift-test-2025-03-11-1 \
  --build-version gamelift-test-2025-03-11-1 \
  --build-root out/windows/amd64/gamelift-servers-managed-ec2 \
  --operating-system WINDOWS_2016 \
```

```
--server-sdk-version 5.3.0 \  
--region us-west-2
```

Note

對於 Mac 和 Linux 組建，請使用 `--operating-system AMAZON_LINUX_2023`

當您建立組建時，請從 API 回應記錄組建 ID，以將其用於建立機群。

步驟 5：建立受管 EC2 機群

下列步驟說明最小機群組態，因此您可以盡快啟動並執行。

若要建立您的機群：

1. 登入 AWS Management Console 並導覽至 Amazon GameLift Servers。
2. 在主控制台視窗頂端的選單列上，檢查以確保您的組建位於哪個區域。請注意，因為您的機群必須位於相同區域，否則您將無法尋找或選擇您的組建。
3. 在受管 EC2 區段的導覽窗格中，選擇組建。
4. 選取您先前上傳的組建，以顯示組建詳細資訊頁面。
5. 在機群區段中，選擇建立機群，該機群會顯示定義受管 EC2 機群詳細資訊頁面，您可以從中追蹤機群的狀態，並在事件索引標籤上檢視機群建立事件。
6. 填寫名稱和描述，然後選擇下一步。
7. 在定義執行個體詳細資訊頁面中，預設會顯示組建的區域。選擇您要新增的任何其他區域。
8. 針對機群類型，選擇隨需。
9. 在執行個體類型下，選擇 `c5.large`，然後選擇下一步。
10. 在執行期組態下，由於您上傳的遊戲組建使用包裝函式，因此您需要改為指向包裝函式可執行檔。對於 Windows 遊戲伺服器，這是 `C:\game\amazon-gamelift-servers-game-server-wrapper.exe`。對於 Linux 遊戲伺服器，這是 `/local/game/amazon-gamelift-servers-game-server-wrapper`。

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

此外，使用允許在 `config.yaml` 和執行時間組態啟動參數中設定的連接埠的範圍來設定遊戲連接埠值。中的連接埠 `config.yaml` 不需要符合機群執行時間組態中指定的連接埠，但如果組態指定不同的連接埠，則該值會覆寫 中的內容 `config.yaml`。在執行時間組態中輸入的啟動參數也會覆寫 中的內容 `config.yaml`。

11. 在檢閱和建立頁面上，再次檢查所有組態，然後選擇提交以建立您的機群。機群狀態會在啟動容量以託管遊戲伺服器時變更，而且很快就會顯示作用中狀態。啟用完成並部署機群後，服務會啟動包裝函式，準備好接收遊戲工作階段請求。

步驟 6：建立並連線至遊戲工作階段

當您的機群狀態顯示作用中時，這表示遊戲伺服器已準備好並等待託管遊戲工作階段。若要啟動遊戲工作階段，請向 Amazon GameLift Servers 服務提出遊戲工作階段請求。在這裡，您將使用 AWS CLI 提出此請求。

Note

請記住，使用 建立遊戲工作階段對於測試和熟悉程序 AWS CLI 非常有用。在某些時間點，您會將程式設計 AWS SDK 呼叫新增至遊戲後端服務，做為遊戲配對或遊戲工作階段放置系統的一部分。

使用下列項目來建立遊戲工作階段：

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

您也可以將自訂遊戲屬性傳遞至伺服器可執行檔。如需詳細資訊，請參閱 README 中的遊戲伺服器引數。當它收到 `create-game-session` 呼叫時，會 Amazon GameLift Servers 通知包裝函式啟動遊戲伺服器可執行檔並啟動遊戲工作階段。中的內容 `config.yaml` 會影響遊戲伺服器的組態，而主控台中設定的啟動參數會決定遊戲工作階段本身的組態。

新增遊戲屬性的範例格式：

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

步驟 7：管理和監控您的機群

現在已設定遊戲伺服器機群並啟動遊戲工作階段，您可以從Amazon GameLift Servers主控台管理和監控它。完成此操作的最佳方法是從機群詳細資訊頁面，您可以在擴展索引標籤上編輯機群詳細資訊或變更機群擴展和容量。請參閱下列有關擴展遊戲伺服器的章節。

選擇指標索引標籤，以檢視描述活動和硬體指標監控的圖表。如需指標圖表的詳細資訊，請選擇機群 ID 旁的資訊連結。此外，雖然您可以從指標索引標籤密切監控遊戲伺服器，您也可以可以在 CloudWatch 儀表板中將警示新增至這些指標。

從主控台導覽至 CloudWatch 儀表板：

1. 在搜尋列中輸入「CloudWatch」，然後從搜尋結果清單中選擇它，以顯示 CloudWatch 概觀。
2. 向下捲動並選擇檢視 GameLift 儀表板，根據機群和遊戲工作階段的關鍵指標來查看圖形。

步驟 8：擴展您的遊戲伺服器

對於下一個步驟，您可以設定自動擴展。透過自動擴展，您可以動態擴展機群容量以回應遊戲伺服器活動。隨著玩家抵達和開始遊戲工作階段，自動擴展會新增更多執行個體，隨著玩家需求減少，自動擴展會釋出未使用的執行個體。這是將託管資源和成本降至最低的有效方法，同時仍提供流暢、快速的玩家體驗。

準備啟動遊戲時，建議您為機群設定自動擴展。建議您使用自動擴展作為有效方式，將託管資源和成本降至最低，同時仍提供順暢、快速的玩家體驗。

手動設定機群容量

手動設定機群容量

1. 前往機群詳細資訊頁面中的擴展索引標籤。
2. 選取位置，然後選擇編輯。
3. 變更所需的執行個體值，並變更大小下限和上限設定，以擴展超過其目前值，然後選擇確認。

Note

使用最大執行個體設定作為停止間隙，以防止過度擴展和成本。

使用目標型自動擴展

使用目標型自動擴展

目標型自動擴展（目標追蹤）會將機群擴展連結至可用遊戲工作階段的百分比。隨著玩家玩遊戲的激增和可用的遊戲工作階段減少，它會透過自動新增更多執行個體到機群來回應。

1. 在目標型自動擴展政策下，選擇新增政策，並將機群容量設定為在達到您設定之可用遊戲工作階段百分比的閾值時自動變更。較大的緩衝區可以更好地處理突增，讓新玩家快速進入遊戲，但也可能導致更高的託管成本。
2. 選擇確認以接受變更。

規則型自動擴展可讓您更精細地控制，例如能夠將擴展連結到其他機群指標，並設定自訂閾值和擴展回應。它提供強大的選項，但還需要使用 CLI 和重大測試來了解自訂規則的運作方式。本教學課程著重於先設定目標型方法。

對常見問題進行故障診斷

以下是您在讓遊戲伺服器和遊戲工作階段執行時可能遇到的常見問題。當伺服器或遊戲工作階段未正確執行時，第一步是檢查日誌，這可能會顯示生產中新部署或遊戲的下列其中一個問題。

日誌中通常會顯示以下內容：

- 遊戲伺服器程序無法啟動。這可能是包裝函式組態中的錯誤——確認檔案具有正確的啟動路徑和正確的啟動參數和引數。

- 遊戲伺服器建置無法執行。這可能是遊戲程式碼中的錯誤。
- 玩家無法連線到遊戲工作階段。這可能是連接埠組態錯誤。
- 延遲或慢速連線。檢閱擴展政策和閾值。
- 無連線。驗證機群的連接埠規則和組態。

檢視機Amazon GameLift Servers群的事件日誌

檢視機Amazon GameLift Servers群的事件日誌

1. 開啟 Amazon GameLift Servers 主控台。
2. 在機群詳細資訊頁面上，選擇事件索引標籤並下載日誌。您也可以從指標索引標籤監控遊戲伺服器運作狀態和遊戲工作階段啟用的活動和硬體指標。

檢視遊戲工作階段日誌

檢視遊戲工作階段日誌

1. 從主控台開啟您的機群，並開啟遊戲工作階段索引標籤。
2. 從清單中選擇遊戲工作階段 ID，以顯示其概觀頁面。
3. 選擇下載日誌以在本機下載日誌檔案。

若要使用 CLI 檢視遊戲工作階段日誌，請使用 `GetGameSessionLogURL` API。Amazon GameLift Servers 會自動儲存日誌 14 天。

您也可以為機群設定 Amazon CloudWatch logs。這提供額外的記錄功能，並與其他 AWS 監控服務整合。

對於透過 CloudWatch 的即時日誌存取或延長保留期間：

1. 在 Amazon GameLift Servers 主控台儀表板頂端的搜尋列中輸入「CloudWatch」，然後從結果下拉式清單中選擇它。
2. 前往 CloudWatch Log Groups 並搜尋特定工作階段。最簡單的方法是按一下全部搜尋並使用 `gameSessionId` 或 `clientId` 進行篩選。

後續步驟

- [建立多位置機群，以在其他位置新增託管](#)
- [新增遊戲工作階段佇列，為跨多個位置的遊戲工作階段提供最佳配置](#)
- [透過為您的遊戲建立配對建構器和規則集來實驗 FlexMatch 配對](#)
- [開始為您的遊戲用戶端和後端服務元件使用功能，讓玩家可以提出加入請求並直接連線至遊戲工作階段](#)
- [準備就緒時，請移至完全整合的解決方案](#)

使用 Amazon GameLift Servers 外掛程式探索

Amazon GameLift Servers 外掛程式是 Unreal 或 Unity 遊戲引擎的完整功能附加元件。它會引導您完成部署遊戲以使用託管的基本步驟Amazon GameLift Servers。使用外掛程式的工具集和工作流程，您可以在遊戲引擎開發環境中工作，準備您的遊戲伺服器進行託管、在本機機器上設定託管進行測試、建立簡單的後端服務，以及將遊戲伺服器部署到受管雲端託管。

使用 外掛程式來體驗使用 Amazon GameLift Servers，並快速啟動和執行遊戲託管解決方案。您可以使用範例遊戲資產或您自己的遊戲專案。外掛程式會自動化許多步驟，讓您可以快速建置簡單的工作解決方案。當您完成外掛程式的引導式工作流程時，您將能夠透過 將遊戲用戶端連線至即時託管遊戲工作階段Amazon GameLift Servers。使用 外掛程式建立簡單的託管解決方案後，您可以自訂解決方案，以滿足遊戲的需求。

外掛程式適用於下列遊戲引擎：

- 虛擬引擎
- Unity

外掛程式包含每個遊戲引擎的這些元件：

- 遊戲引擎編輯器的外掛程式模組。安裝外掛程式時，新的主功能表按鈕可讓您存取 Amazon GameLift Servers 功能。
- 具有用戶端功能的 Amazon GameLift Servers 服務 API 程式庫。
- Amazon GameLift Servers 伺服器 SDK（第 5 版）的程式庫。
- 用於測試伺服器整合的範例資產。
- 可編輯組態，採用 AWS CloudFormation 範本形式，可定義您的遊戲伺服器解決方案。

主題

- [外掛程式工作流程](#)
- [Amazon GameLift ServersUnreal Engine 的外掛程式](#)
- [Amazon GameLift ServersUnity 的外掛程式 \(伺服器 SDK 5.x\)](#)
- [Amazon GameLift ServersUnity 的外掛程式 \(伺服器 SDK 4.x\)](#)

外掛程式工作流程

下列步驟說明在 上準備和部署遊戲專案的典型路徑Amazon GameLift Servers。您可以在遊戲引擎編輯器和遊戲程式碼中操作，以完成這些步驟。

1. 建立使用者描述檔，以連結至 AWS 您的帳戶使用者，並提供存取登入資料使用的許可Amazon GameLift Servers。
2. 設定外掛程式在託管解決方案中使用的相關 AWS 資源（稱為「引導」）。
3. 將伺服器程式碼新增至您的專案，以在執行中的遊戲伺服器和服務之間建立通訊Amazon GameLift Servers。
4. 將用戶端程式碼新增至您的專案，讓遊戲用戶端將請求傳送至 Amazon GameLift Servers以啟動新的遊戲工作階段，然後連線到它們。
5. 使用 Anywhere 工作流程將本機工作站設定為 Anywhere 運算和託管遊戲伺服器。透過外掛程式在本機啟動遊戲伺服器和用戶端、連線至遊戲工作階段，並測試整合。
6. 使用 受管 EC2 工作流程將您的遊戲伺服器上傳至 ，Amazon GameLift Servers並部署簡單但完整的雲端託管解決方案。透過外掛程式在本機啟動遊戲用戶端、請求遊戲工作階段並與其連線，以及玩遊戲。

在外掛程式中工作時，您將建立和使用 AWS 資源，這些動作可能會對使用中的 AWS 帳戶產生費用。如果您是初次使用 AWS，這些動作可能涵蓋在 [AWS 免費方案](#)中。

Amazon GameLift ServersUnreal Engine 的外掛程式

此外掛程式會將 Amazon GameLift Servers C++ 伺服器 SDK 和工具新增至 UE 編輯器。使用引導式 UI 工作流程將伺服器 SDK 功能整合到您的遊戲專案中，並為您的遊戲伺服器部署Amazon GameLift Servers託管解決方案。

使用 外掛程式，您可以建置基本的工作託管解決方案，然後視需要最佳化和自訂。使用本機工作站做為主機來設定 Amazon GameLift Servers Anywhere 機群。對於使用受管 EC2 或受管容器機群的雲端託管，請使用完整的解決方案部署遊戲伺服器，以管理遊戲工作階段請求和用戶端連線。

主題

- [安裝 Unreal 遊戲專案的外掛程式](#)
- [後續步驟：自訂您的遊戲託管解決方案](#)
- [Unreal 的外掛程式：設定 AWS 使用者設定檔](#)
- [Unreal 的外掛程式：整合您的遊戲程式碼](#)
- [Unreal 的外掛程式：使用 Amazon GameLift ServersAnywhere 在本機託管您的遊戲](#)
- [Unreal 的外掛程式：將您的遊戲部署到受管 EC2 機群](#)
- [Unreal 的外掛程式：將您的遊戲部署到受管容器機群](#)

安裝 Unreal 遊戲專案的外掛程式

[從 GitHub 取得 Unreal Engine 的 Amazon GameLift Servers 外掛程式](#)

如需如何在遊戲專案的 Unreal Editor 中安裝外掛程式的相關資訊，請參閱 GitHub 儲存庫讀我檔案。

外掛程式包含下列元件：

- UE 編輯器的外掛程式模組。安裝外掛程式時，新的主功能表按鈕可讓您存取 Amazon GameLift Servers 功能。
- Amazon GameLift Servers 服務 API 的 C++ 程式庫。在用戶端後端服務中使用 API 功能，以協助遊戲用戶端請求遊戲工作階段和傳送/擷取遊戲工作階段資訊。
- Amazon GameLift Servers 伺服器 SDK（第 5 版）的不真實程式庫。在您的遊戲伺服器程式碼中使用伺服器 SDK 來管理託管遊戲伺服器程序與服務之間的通訊 Amazon GameLift Servers。
- 用於測試的內容，包括啟動遊戲地圖和兩個具有基本藍圖的測試地圖，以及用於測試伺服器整合的 UI 元素。
- 可編輯組態，採用 AWS CloudFormation 範本形式，外掛程式會在部署遊戲伺服器進行託管時使用。

此外掛程式使用 AWS CloudFormation 範本來部署常見遊戲案例的託管解決方案。依照提供的方式使用這些解決方案，或視需要為您的遊戲自訂解決方案。

後續步驟：自訂您的遊戲託管解決方案

使用外掛程式的引導式工作流程是使用 Amazon GameLift Servers 託管解決方案快速啟動和執行的好方法。使用 外掛程式，您可以設定每個解決方案元件的基本版本。

當您準備好時，您可以透過自訂每個元件來建置此基本解決方案，並在準備遊戲啟動時微調您的解決方案。請考慮以下選項：

- 修改您的機群和機群組態。請參閱 [自訂您的 Amazon GameLift Servers EC2 受管機群](#)。
- 自訂您的遊戲工作階段佇列組態。請參閱 [自訂遊戲工作階段佇列](#)：
- 將功能新增至遊戲伺服器與遊戲用戶端。請參閱 [將遊戲伺服器與整合 Amazon GameLift Servers](#) 和 [將您的遊戲用戶端與整合 Amazon GameLift Servers](#)。
- 自訂您的後端服務。請參閱 [設計您的遊戲用戶端服務](#)。
- 設定自動容量擴展以滿足預期的玩家需求。請參閱 [使用擴展遊戲託管容量 Amazon GameLift Servers](#)。
- 設定託管可觀測性工具，包括分析和記錄。請參閱 [監控 Amazon GameLift Servers](#)。
- 使用[基礎設施即程式碼 \(IaC\)](#) 自動化部署。外掛程式的受管解決方案引導工作流程使用 AWS CloudFormation 範本。您可以視需要自訂這些項目。請參閱 [使用管理 Amazon GameLift Servers 託管資源 AWS CloudFormation](#)。

主題

- [Unreal 的外掛程式：設定 AWS 使用者設定檔](#)
- [Unreal 的外掛程式：整合您的遊戲程式碼](#)
- [Unreal 的外掛程式：使用 Amazon GameLift ServersAnywhere 在本機託管您的遊戲](#)
- [Unreal 的外掛程式：將您的遊戲部署到受管 EC2 機群](#)
- [Unreal 的外掛程式：將您的遊戲部署到受管容器機群](#)

Unreal 的外掛程式：設定 AWS 使用者設定檔

安裝外掛程式後，請使用有效的 AWS 帳戶設定使用者設定檔。您可以在外掛程式中維護多個設定檔，但一次只能選取一個設定檔。每當您在外掛程式中工作時，請選取要使用的設定檔。每個工作流程頁面會顯示目前選取的設定檔。

維護多個設定檔可讓您在不同託管部署之間切換。例如，您可以設定使用相同 AWS 帳戶但部署到不同 AWS 區域的設定檔。或者，您可以使用不同的 AWS 帳戶或使用者和許可集來設定設定檔。

Note

如果您已在工作站上安裝 AWS CLI 並已設定設定檔，Amazon GameLift Servers 外掛程式會偵測它並將其列為現有的設定檔。外掛程式會自動選取任何名為 `default` 的設定檔。您可以使用現有的設定檔或建立新的設定檔。

必須引導所有設定檔，才能在您的帳戶使用者下設定一些必要的 AWS 資源。

管理您的 AWS 設定檔

1. 在 Unreal 編輯器主工具列中，選擇 Amazon GameLift Servers 選單，然後選取 AWS 存取登入資料。此動作會將 Amazon GameLift Servers 外掛程式開啟至頁面設定您的使用者設定檔。
2. 使用按鈕來建立新的 AWS 帳戶，或為您已有 AWS 的帳戶設定使用者設定檔。
3. 如果您還沒有使用者設定檔，系統會提示您輸入設定檔詳細資訊並建立新的設定檔。請提供下列資訊：
 - AWS 帳戶。如果您建立新 AWS 帳戶，請使用連結 AWS Management Console 並依照提示操作。如需詳細資訊，[請參閱建立 AWS 帳戶](#)。
 - 具有使用 Amazon GameLift Servers 和其他必要 AWS 服務許可 AWS 的使用者。[設定 AWS 帳戶](#) 如需設定具有 Amazon GameLift Servers 許可的 AWS Identity and Access Management (IAM) 使用者，以及使用長期憑證進行程式設計存取的指示，請參閱。
 - 您 AWS 使用者的登入資料。這些登入資料包含 AWS 存取金鑰 ID 和 AWS 私密金鑰。如需詳細資訊，請參閱[取得您的存取金鑰](#)。
 - AWS 區域。這是您要建立 AWS 資源以進行託管的地理位置。在開發期間，我們建議您使用靠近實體位置的區域。從[支援的區域清單中選擇 AWS 區域](#)。
4. 如果外掛程式偵測到現有的設定檔，則會顯示可用的設定檔清單。從清單中選擇現有的設定檔，或選擇新增另一個設定檔以建立新的設定檔。

引導使用者設定檔

所有設定檔都必須引導才能與 Amazon GameLift Servers 外掛程式搭配使用。引導會建立設定檔專屬的 Amazon S3 儲存貯體。它用於存放專案組態、建置成品和其他相依性。儲存貯體不會在其他設定檔之間共用。

引導涉及建立新的 AWS 資源，並可能產生成本。

若要引導您的設定檔：

1. 在AWS 存取登入資料頁面上，檢查您要使用的使用者設定檔的引導狀態。如果設定檔的引導狀態為「非作用中」且未列出 S3 儲存貯體，則需要引導設定檔。
2. 選取您要使用的設定檔，然後選擇引導設定檔。
3. 等待引導狀態變更為「作用中」。這可能需要幾分鐘的時間。

Unreal 的外掛程式：整合您的遊戲程式碼

在您可以將遊戲伺服器部署到機群之前，您需要對遊戲程式碼和套件遊戲元件進行一系列更新，以便與服務搭配使用Amazon GameLift Servers。

本主題會逐步解說進行最少整合的步驟。對於伺服器整合，請使用提供的程式碼範例來更新專案的遊戲模式。

- [設定建置目標和模組規則](#)
- [更新您的遊戲伺服器程式碼](#)
- [整合您的用戶端遊戲地圖](#)
- [封裝您的遊戲元件](#)

設定建置目標和模組規則

修改遊戲專案檔案以正確產生建置元件，以搭配 使用Amazon GameLift Servers。

若要新增用戶端和伺服器建置目標：

1. 開啟遊戲專案的程式碼檔案，並找到 `.../Games/[your application name]Source/[your application name]Target.cs` 檔案。範例：`.../Source/GameLiftUnrealAppTarget.cs`。（如果您使用 Visual Studio，請開啟專案.sln的檔案。）
2. 複製此檔案以在 Source/目錄中建立新的兩個目標檔案。
 - 用戶端目標 – 將新檔案重新命名為 `[your application name]Client.Target.cs`。編輯內容以更新類別名稱和目標類型值，如下列範例程式碼所示：

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

public class GameLiftUnrealAppClientTarget : TargetRules
```

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

- 伺服器目標 – 將新檔案重新命名為 *[your application name]*Server.Target.cs。編輯內容以更新類別名稱和目標類型值，如下列範例程式碼所示：

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

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

3. 重新產生專案檔案。如果您使用的是 Visual Studio，您可以在遊戲專案的 .uproject 檔案上按一下滑鼠右鍵，然後選取產生 Visual Studio 專案檔案。

若要更新遊戲專案模組規則：

更新遊戲專案的模組規則，以依賴外掛程式。

1. 開啟遊戲專案的程式碼檔案，並找到 .../Games/*[your application name]*Source/*[your application name]*.Build.cs 檔案。範例：.../Source/GameLiftUnrealApp.Build.cs。（如果您使用 Visual Studio，請開啟專案.sln的檔案。）
2. 找到 ModuleRules類別並更新，如下列範例程式碼所示：

```
using UnrealBuildTool;
```



```

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

```

3. 建立新的目標檔案並修改模組規則後，請重建您的遊戲專案。

更新您的遊戲伺服器程式碼

更新您的遊戲伺服器程式碼，以啟用遊戲伺服器程序與服務之間的通訊Amazon GameLift Servers。您的遊戲伺服器必須能夠回應來自的請求Amazon GameLift Servers，例如啟動和停止新的遊戲工作階段。

新增的伺服器程式碼 Amazon GameLift Servers

1. 在程式碼編輯器中，開啟遊戲專案的解決方案 (.sln) 檔案，通常位於專案根資料夾中。例如：GameLiftUnrealApp.sln。
2. 開啟解決方案後，找到專案遊戲模式標頭檔案：[project-name]GameMode.h 檔案。例如：GameLiftUnrealAppGameMode.h。
3. 變更標頭檔案以符合下列程式碼。請務必將 "GameLiftServer" 取代為您自己的專案名稱。這些更新是遊戲伺服器特有的；我們建議您備份原始遊戲模式檔案，以便與用戶端搭配使用。

gameMode.h 程式碼範例

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

#pragma once

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

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

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

public:
    AGameLiftUnrealAppGameMode();

protected:
    virtual void BeginPlay() override;

private:
    void InitGameLift();

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

- 開啟相關的來源[project-name]GameMode.cpp檔案（例如 GameLiftUnrealAppGameMode.cpp）。變更程式碼以符合下列範例程式碼。請務必將 "GameLiftUnrealApp" 取代為您自己的專案名稱。這些更新是遊戲伺服器特有的；我們建議您備份原始檔案，以便與用戶端搭配使用。

下列範例程式碼說明如何新增與 進行伺服器整合所需的最低元素Amazon GameLift Servers：

- 初始化 Amazon GameLift Servers API 用戶端。Amazon GameLift Servers Anywhere 機群需要具有伺服器參數的 InitSDK() 呼叫。當您連線到 Anywhere 機群時，外掛程式會將伺服器參數儲存為主控台引數。範例程式碼可在執行時間存取這些值。
- 實作必要的回呼函數，以回應來自 Amazon GameLift Servers 服務的請求，包括 OnStartGameSession、OnProcessTerminate 和 onHealthCheck。
- ProcessReady() 使用指定的連接埠呼叫，以便在準備好託管遊戲工作階段時通知 Amazon GameLift Servers 服務。

遊戲伺服器程式碼範例

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

#include "GameLiftUnrealAppGameMode.h"

#include "UObject/ConstructorHelpers.h"
#include "Kismet/GameplayStatics.h"

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

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

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

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

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

```

}

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

#if WITH_GAMELIFT
    InitGameLift();
#endif
}

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

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

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

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

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

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

```

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

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

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

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

FString glAnywhereAwsRegion = "";
if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAwsRegion="),
glAnywhereAwsRegion))
{

```

```

        ServerParametersForAnywhere.m_awsRegion =
TCHAR_TO_UTF8(*glAnywhereAwsRegion);
    }

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

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

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

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

```

```

        UE_LOG(GameServerLog, Log, TEXT(">>>> Secret Key: %s"),
*ServerParametersForAnywhere.m_secretKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Session Token: %s"),
*ServerParametersForAnywhere.m_sessionToken);
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }

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

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

    ProcessParameters = MakeShared<FProcessParameters>();

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

```



```

});

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

//This is the HealthCheck callback.
//Amazon GameLift Servers will invoke this callback every 60 seconds or so.
//Here, a game server might want to check the health of dependencies and such.
//Simply return true if healthy, false otherwise.
//The game server has 60 seconds to respond with its health status. Amazon GameLift
Servers will default to 'false' if the game server doesn't respond in time.
//In this case, we're always healthy!

```

```
ProcessParameters->OnHealthCheck.BindLambda([]()  
{  
    UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));  
    return true;  
});  
  
//GameServer.exe -port=7777 LOG=server.mylog  
ProcessParameters->port = FURL::UrlConfig.DefaultPort;  
TArray<FString> CommandLineTokens;  
TArray<FString> CommandLineSwitches;  
  
FCommandLine::Parse(FCommandLine::Get(), CommandLineTokens, CommandLineSwitches);  
  
for (FString SwitchStr : CommandLineSwitches)  
{  
    FString Key;  
    FString Value;  
  
    if (SwitchStr.Split("=", &Key, &Value))  
    {  
        if (Key.Equals("port"))  
        {  
            ProcessParameters->port = FCString::Atoi(*Value);  
        }  
    }  
}  
  
//Here, the game server tells Amazon GameLift Servers where to find game session  
log files.  
//At the end of a game session, Amazon GameLift Servers uploads everything in the  
specified  
//location and stores it in the cloud for access later.  
TArray<FString> Logfiles;  
Logfiles.Add(TEXT("GameLiftUnrealApp/Saved/Logs/server.log"));  
ProcessParameters->logParameters = Logfiles;  
  
//The game server calls ProcessReady() to tell Amazon GameLift Servers it's ready  
to host game sessions.  
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready..."));  
FGameLiftGenericOutcome ProcessReadyOutcome = GameLiftSdkModule->  
>ProcessReady(*ProcessParameters);  
  
if (ProcessReadyOutcome.IsSuccess())  
{
```

```

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

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

```

整合您的用戶端遊戲地圖

啟動遊戲映射包含藍圖邏輯和 UI 元素，這些元素已包含基本程式碼來請求遊戲工作階段，並使用連線資訊來連線至遊戲工作階段。您可以照原樣使用映射，或視需要修改這些映射。將啟動遊戲地圖與其他遊戲資產搭配使用，例如 Unreal Engine 提供的第三方範本專案。這些資產可在內容瀏覽器中使用。您可以使用它們來測試外掛程式的部署工作流程，或做為為您的遊戲建立自訂後端服務的指南。

啟動映射具有下列特性：

- 它包含適用於 Anywhere 機群和受管 EC2 機群的邏輯。當您執行用戶端時，您可以選擇連線到任一機群。
- 用戶端功能包括尋找遊戲工作階段 (`SearchGameSessions()`)、建立新的遊戲工作階段 (`CreateGameSession()`)，以及直接加入遊戲工作階段。
- 它會從專案的 Amazon Cognito 使用者集區取得唯一的玩家 ID（這是已部署 Anywhere 解決方案的一部分）。

使用啟動遊戲地圖

1. 在 UE 編輯器中，開啟專案設定、映射和模式頁面，然後展開預設映射區段。
2. 對於編輯器啟動映射，從下拉式清單中選取「StartupMap」。您可能需要搜尋位於的檔案...
> Unreal Projects/[project-name]/Plugins/Amazon GameLift Servers Plugin Content/Maps。

3. 對於遊戲預設映射，從下拉式清單中選取相同的「StartupMap」。
4. 針對伺服器預設映射，選取「ThirdPersonMap」。這是遊戲專案中包含的預設地圖。此地圖專為遊戲中的兩個玩家而設計。
5. 開啟伺服器預設映射的詳細資訊面板。將 GameMode 覆寫設定為「無」。
6. 展開預設模式區段，並將全域預設伺服器遊戲模式設定為您為伺服器整合更新的遊戲模式。

對專案進行這些變更之後，您就可以建置遊戲元件。

封裝您的遊戲元件

封裝遊戲伺服器和遊戲用戶端組建

1. 在 Unreal Engine 編輯器的來源建置版本中開啟您的遊戲專案。
2. 使用編輯器封裝遊戲用戶端和伺服器組建。
 - a. 選擇目標。前往平台、Windows 並選取下列其中一項：
 - 伺服器：[your-application-name]Server
 - 用戶端：[your-application-name]Client
 - b. 啟動建置。前往平台、Windows、套件專案。

每個封裝程序都會產生可執行檔：[your-application-name]Client.exe 或 [your-application-name]Server.exe。

在 外掛程式中，設定本機工作站上用戶端和伺服器建置可執行檔的路徑。

Unreal 的外掛程式：使用 Amazon GameLift Servers Anywhere 在本機託管您的遊戲

使用此工作流程，使用 Anywhere 機群將本機工作站設定為遊戲伺服器主機。您可以使用它來測試遊戲伺服器整合，然後再部署到雲端型受管機群。它也適用於反覆進行遊戲開發期間的本機測試。

若要啟動 Amazon GameLift Servers Anywhere 工作流程：

- 在 Unreal 編輯器主工具列中，選擇 Amazon GameLift Servers 選單，然後選取 Host with Anywhere。此動作會開啟外掛程式頁面 Deploy Anywhere，它提供六個步驟來整合、建置和啟動遊戲元件。

步驟 1：設定您的設定檔

選擇您要在遵循此工作流程時使用的設定檔。您選取的設定檔會影響工作流程中的所有步驟。您建立的所有資源都會與設定檔 AWS 的帳戶相關聯，並放置在設定檔的預設 AWS 區域中。設定檔使用者的許可決定您對 AWS 資源和動作的存取。

設定使用者設定檔

1. 從可用設定檔的下拉式清單中選取設定檔。如果您還沒有設定檔，或想要建立新的設定檔，請前往 Amazon GameLift 功能表，然後選擇設定 AWS 使用者設定檔。
2. 如果引導狀態不是「作用中」，請選擇引導設定檔，並等待狀態變更為「作用中」。

步驟 2：設定您的遊戲程式碼

在此步驟中，準備您的遊戲伺服器與遊戲用戶端組建以使用 Amazon GameLift Servers。如果您尚未整合遊戲程式碼，請參閱 [Unreal 的外掛程式：整合您的遊戲程式碼](#)。輸入本機工作站上遊戲可執行檔的路徑。

- 遊戲伺服器：將您的遊戲伺服器與適用於的伺服器 SDK 整合，Amazon GameLift Servers 並封裝您的遊戲伺服器建置。如需說明，請參閱 [Unreal 的外掛程式：整合您的遊戲程式碼](#)。遊戲伺服器必須與伺服器 SDK 整合，以建立與服務的通訊 Amazon GameLift Servers，並回應啟動新遊戲工作階段和接受遊戲用戶端連線的提示。
- 遊戲用戶端：您至少需要能夠使用 IP 地址和連接埠資訊連線到遊戲伺服器的遊戲用戶端。如果您尚未為設定遊戲用戶端元件 Amazon GameLift Servers，您可以使用 AWS CLI 工具手動請求新的遊戲工作階段、取得連線資訊，並使用該資訊來連接遊戲用戶端。

在某些時候，您需要有後端服務，才能將新的遊戲工作階段請求傳送至 Amazon GameLift Servers 服務，並將連線資訊轉傳回遊戲用戶端。您可以使用外掛程式隨附的測試地圖，將用戶端 Amazon GameLift Servers 功能新增至您的遊戲專案。如需建置自訂解決方案的說明，請參閱 [Amazon GameLift Servers 新增至您的遊戲用戶端](#)。

步驟 3：連線至 Anywhere 機群

在此步驟中，您會指定要使用的 Anywhere 機群。Anywhere 機群定義了運算資源的集合，這些資源可以位於任何地方，用於遊戲伺服器託管。

- 如果您目前正在使用的 AWS 帳戶有現有的 Anywhere 機群，請開啟機群名稱下拉式清單欄位，然後選擇機群。此下拉式清單只會顯示目前作用中使用者設定檔 AWS 區域中的 Anywhere 機群。

- 如果沒有現有機群，或您想要建立新的機群，請選擇建立新的 Anywhere 機群並提供機群名稱。

在您為專案選擇 Anywhere 機群之後，Amazon GameLift Servers 會驗證機群狀態為作用中，並顯示機群 ID。您可以在 Unreal 編輯器的輸出日誌中追蹤此請求的進度。

步驟 4：註冊您的工作站

在此步驟中，您會將本機工作站註冊為新 Anywhere 機群中的運算資源。

將工作站註冊為 Anywhere 運算

1. 輸入本機電腦的運算名稱。如果您在機群中新增多個運算，名稱必須是唯一的。
2. 為您的本機電腦提供 IP 地址。此欄位預設為您機器的公有 IP 地址。您也可以使用 localhost (127.0.0.1)，只要您在相同的電腦上執行遊戲用戶端和伺服器。
3. 選擇註冊運算。您可以在 Unreal 編輯器的輸出日誌中追蹤此請求的進度。

為了回應此動作，Amazon GameLift Servers 會驗證它是否可以連接到運算，並傳回有關新註冊的運算的資訊。它也會建立遊戲可執行檔初始化與服務通訊時所需的主控台引數 Amazon GameLift Servers。

步驟 5：產生身分驗證字符

在 Anywhere 運算上執行的遊戲伺服器程序需要身分驗證字符才能呼叫 Amazon GameLift Servers 服務。每當您從外掛程式啟動遊戲伺服器時，外掛程式會自動產生並存放 Anywhere 機群的身分驗證權杖。驗證字符值會儲存為命令列引數，您的伺服器程式碼可在執行時間擷取該引數。

上述程式碼範例也可讓您將 [AWS Signature 第 4 版 \(SigV4\) 用於 API 請求](#)。SigV4 是將身分驗證資訊新增至 API 請求的 AWS 簽署通訊協定。

在此步驟中，您不需要採取任何動作。

步驟 6：啟動遊戲

此時，您已完成使用 在本機工作站上啟動和播放多玩家遊戲所需的所有任務 Amazon GameLift Servers。

播放您的託管遊戲

1. 啟動您的遊戲伺服器。遊戲伺服器將在準備好託管遊戲工作階段 Amazon GameLift Servers 時通知。

2. 啟動您的遊戲用戶端並使用新功能來啟動新的遊戲工作階段。此請求會透過新的後端服務傳送至 Amazon GameLift Servers。為了回應，會 Amazon GameLift Servers 呼叫在本機電腦上執行的遊戲伺服器，以啟動新的遊戲工作階段。當遊戲工作階段準備好接受玩家時，Amazon GameLift Servers 會提供連線資訊，讓遊戲用戶端加入遊戲工作階段。

Unreal 的外掛程式：將您的遊戲部署到受管 EC2 機群

在此工作流程中，部署您的遊戲以託管管理的雲端型運算資源 Amazon GameLift Servers。將您的整合遊戲伺服器建置上傳至 Amazon GameLift Servers 服務以進行部署。如果您尚未整合遊戲程式碼，請參閱 [Unreal 的外掛程式：整合您的遊戲程式碼](#)。當此工作流程完成時，您會有一個運作中的遊戲用戶端，可以連線到雲端中的遊戲伺服器。

若要啟動 Amazon GameLift Servers 受管 Amazon EC2 工作流程：

- 在 Unreal 編輯器主工具列中，選擇 Amazon GameLift Servers 選單，然後選取 Host with Managed EC2。此動作會開啟外掛程式頁面部署 Amazon EC2 機群，此機群提供六個步驟來整合、建置、部署和啟動遊戲元件。

步驟 1：設定您的設定檔

選擇您要在遵循此工作流程時使用的設定檔。您選取的設定檔會影響工作流程中的所有步驟。您建立的所有資源都會與設定檔 AWS 的帳戶相關聯，並放置在設定檔的預設 AWS 區域中。設定檔使用者的許可決定您對 AWS 資源和動作的存取。

設定使用者設定檔

1. 從可用設定檔的下拉式清單中選取設定檔。如果您還沒有設定檔或想要建立新的設定檔，請前往 Amazon GameLift 功能表，然後選擇設定 AWS 使用者設定檔。
2. 如果引導狀態不是「作用中」，請選擇引導設定檔，並等待狀態變更為「作用中」。

步驟 2：設定您的遊戲程式碼

在此步驟中，準備您的遊戲伺服器和遊戲用戶端組建，以使用適用於 Unreal 的 Amazon GameLift Servers C++ 伺服器開發套件。如果您尚未整合遊戲程式碼和建置遊戲用戶端和伺服器可執行檔，請參閱 [Unreal 的外掛程式：整合您的遊戲程式碼](#)。輸入本機工作站上遊戲可執行檔的路徑。

工作流程中的此步驟，外掛程式提供指示和原始程式碼的連結，用於設定 Unreal Editor 的來源建置版本。在建置用戶端和伺服器元件時，您需要使用來源建置的版本。

建置與伺服器 SDK 整合的遊戲伺服器後，請完成下列任務，以準備將其上傳至 Amazon GameLift Servers 進行託管。

準備您的伺服器建置以進行雲端部署 (Windows)

在預設存放伺服器建置檔案的 Unreal 編輯器WindowsServer資料夾中，進行下列新增：

1. 將伺服器建置安裝指令碼複製到 **WindowsServer** 資料夾的根目錄。安裝指令碼包含在外掛程式下載中。尋找 檔案 [project-name]/Plugins/Resources/CloudFormation/extra_server_resources/install.bat。Amazon GameLift Servers 使用此檔案將伺服器建置安裝到您的託管運算。
2. 將 **VC_redist.x64.exe** 檔案複製到 **WindowsServer** 資料夾的根目錄。如果您使用的是 Unreal Engine 5.6 版或更新版本，則可以略過此步驟。此檔案包含在 Visual Studio 安裝中。它通常位於 C:/Program Files (x86)/Microsoft Visual Studio/2019/Professional/VC/Redist/MSVC/v142。
3. 將 OpenSSL 程式庫檔案新增至您的遊戲伺服器建置。如果您的遊戲伺服器與伺服器 SDK 5.3 或更新版本整合，您可以略過此步驟。此版本包含在 Unreal 3.0 版或更新版本的 Amazon GameLift Servers 外掛程式中。

手動尋找 OpenSSL 程式庫並將其複製到位於 的遊戲組建套件目錄 <YourGame>/Binaries/Win64。您必須使用與 Unreal Engine 5 版本相同的 OpenSSL 版本。使用錯誤的 OpenSSL 程式庫部署的遊戲組建將無法與服務通訊 Amazon GameLift Servers。

尋找遊戲引擎來源中的 OpenSSL 程式庫。位置會根據您的開發環境而有所不同：

在 Windows 上：

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

在 Linux 上：

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

準備您的伺服器建置以進行雲端部署 (Linux)

如需準備適用於 Linux 的遊戲伺服器的詳細指示，請參閱[在 Amazon Linux 上建置 Amazon GameLift Servers 適用於 Unreal Engine 5 的伺服器 SDK](#)。

1. 指定工作目錄來組織您的建置檔案。工作目錄的結構會依原樣部署到每個託管運算。新增 Linux 建置的遊戲伺服器和所有相依檔案。
2. 在工作目錄的根目錄中建立伺服器建置安裝指令碼。如有需要，請建立 `install.sh` 檔案並新增任何必要命令，以正確安裝遊戲伺服器組建。Amazon GameLift Servers 使用此檔案將伺服器組建安裝到每個 EC2 託管資源。
3. 將 OpenSSL 程式庫檔案新增至您的遊戲伺服器建置。如果您的遊戲伺服器與伺服器 SDK 5.3 或更新版本整合，您可以略過此步驟。

手動尋找和複製程式庫。您必須使用與 Unreal Engine 5 版本相同的 OpenSSL 版本。使用錯誤的 OpenSSL 程式庫部署的遊戲組建將無法與服務通訊 Amazon GameLift Servers。

- a. 尋找遊戲引擎來源中的 OpenSSL 程式庫。位置會根據您的開發環境而有所不同：

在 Windows 上：

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

在 Linux 上：

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

- b. 當您找到 OpenSSL 程式庫時，請將其複製到位於的遊戲組建套件目錄 `<YourGame>/Binaries/Linux`。

步驟 3：選取部署案例

在此步驟中，您可以選擇目前要部署的遊戲託管解決方案。您可以使用任何案例，對遊戲進行多個部署。

- 單一區域機群：將您的遊戲伺服器部署到作用中設定檔預設 AWS 區域中的單一託管資源機群。此案例是測試與 和 伺服器建置組態整合 AWS 的良好起點。它會部署下列資源：
 - AWS 機群（隨需），您的遊戲伺服器建置已安裝並執行。
 - Amazon Cognito 使用者集區和用戶端可讓玩家驗證和啟動遊戲。
 - 將使用者集區與 API 連結APIs 閘道授權方。
 - WebACI 用於調節對 API 閘道的過多玩家呼叫。
 - API 閘道 + Lambda 函數，讓玩家請求遊戲位置。CreateGameSession() 如果沒有可用，此函數會呼叫。
 - API 閘道 + Lambda 函數，讓玩家取得其遊戲請求的連線資訊。
- FlexMatch 機群：將您的遊戲伺服器部署到一組機群，並使用規則設定 FlexMatch 配對建構器來建立玩家配對。此案例使用具有多機群、多位置結構的低成本 Spot 託管，以實現持久的可用性。當您準備好開始為託管解決方案設計配對建構器元件時，此方法非常有用。在此案例中，您將為此解決方案建立基本資源，之後可以視需要自訂。它會部署下列資源：
 - FlexMatch 配對組態和配對規則集，以接受玩家請求並形成配對。
 - 在 AWS 多個位置安裝並執行遊戲伺服器建置的三個機群。包含兩個 Spot 機群和一個隨需機群做為備份。
 - AWS 透過尋找最佳的託管資源（根據可行性、成本、玩家延遲等）並啟動遊戲工作階段，滿足對提議配對的請求的遊戲工作階段置放佇列。
 - Amazon Cognito 使用者集區和用戶端可讓玩家驗證和啟動遊戲。
 - 將使用者集區與 API 連結APIs 閘道授權方。
 - WebACI 用於調節對 API 閘道的過多玩家呼叫。
 - API 閘道 + Lambda 函數，讓玩家請求遊戲位置。此函數會呼叫 StartMatchmaking()。
 - API 閘道 + Lambda 函數，讓玩家取得其遊戲請求的連線資訊。
 - Amazon DynamoDB 資料表可存放玩家和遊戲工作階段資訊的配對票證。
 - SNS 主題 + 處理 GameSessionQueue 事件的 Lambda 函數。

步驟 4：設定遊戲參數

在此步驟中，您會描述要上傳到的遊戲 AWS；

- 伺服器建置名稱：為您的遊戲伺服器建置提供有意義的名稱。AWS 使用此名稱來參考已上傳並用於部署的伺服器建置複本。

- 伺服器建置作業系統：輸入伺服器建置用來執行的作業系統。這會告訴您 AWS 要用來託管遊戲的運算資源類型。
- 遊戲伺服器資料夾：識別本機伺服器建置資料夾的路徑。
- 遊戲伺服器建置：識別遊戲伺服器可執行檔的路徑。
- 遊戲用戶端路徑：識別遊戲用戶端可執行檔的路徑。
- 用戶端組態輸出：此欄位需要指向用戶端建置中包含您 AWS 組態的資料夾。在下列位置尋找它：`[client-build]/[project-name]/Content/CloudFormation`。

步驟 5：部署案例

在此步驟中，您會根據您選擇的部署案例，將遊戲部署至雲端託管解決方案。此程序可能需要幾分鐘的時間，同時 AWS 驗證您的伺服器建置、佈建託管資源、安裝遊戲伺服器、啟動伺服器程序，以及準備好託管遊戲工作階段。

若要開始部署，請選擇部署 CloudFormation。您可以在這裡追蹤遊戲託管的狀態。如需更多詳細資訊，您可以登入的 AWS 管理主控台，並檢視事件通知。請務必使用與外掛程式中作用中使用使用者設定檔相同的帳戶、使用者和 AWS 區域來登入。

部署完成時，您會在 EC2 AWS 執行個體上安裝遊戲伺服器。至少有一個伺服器程序正在執行中，並準備好啟動遊戲工作階段。

步驟 6：啟動用戶端

此時，您已完成啟動和播放使用託管的多玩家遊戲所需的所有任務 Amazon GameLift Servers。若要玩遊戲，請啟動您遊戲用戶端的執行個體。

如果您部署了單一機群案例，則可以使用一個播放器開啟單一用戶端執行個體，輸入伺服器地圖並四處移動。開啟遊戲用戶端的其他執行個體，將第二個玩家新增至相同的伺服器遊戲映射。

如果您部署 FlexMatch 案例，解決方案會等待至少兩個用戶端排入佇列以進行遊戲工作階段放置，然後玩家才能進入伺服器映射。

Unreal 的外掛程式：將您的遊戲部署到受管容器機群

使用此引導式外掛程式工作流程為您的遊戲伺服器建立容器映像，並將其部署至容器型託管解決方案。如果您尚未整合遊戲程式碼，請參閱 [Unreal 的外掛程式：整合您的遊戲程式碼](#)。當您成功完成此工作流程時，您的容器化遊戲伺服器會在雲端執行，而且您可以使用外掛程式來啟動遊戲用戶端、連線至遊戲工作階段，以及玩遊戲。

開始之前

此工作流程假設您已完成下列任務。

- 將您的遊戲伺服器程式碼與Amazon GameLift Servers伺服器 SDK 整合。您的託管遊戲伺服器必須能夠與 Amazon GameLift Servers服務通訊，以便回應啟動新遊戲工作階段和報告遊戲工作階段狀態的請求。如果您尚未完成此任務，建議您先使用 Anywhere 遵循外掛程式工作流程主機。如需準備遊戲伺服器程式碼的指引，請參閱 [更新您的遊戲伺服器程式碼](#)。對於受管容器機群，您必須將遊戲與伺服器 SDK 5.2 版或更新版本整合。

Note

如果您使用的是啟動遊戲地圖，則此任務已為您完成。

- 封裝遊戲伺服器可執行檔以在 Linux 上執行。如果您是在 Windows 上進行開發並整合 C++ 伺服器 SDK 5.2.x 版或更早版本，則需要使用 [Unreal 跨編譯工具組](#)。或者，您可以設定個別的 Linux 工作區，或使用如適用於 Linux 的 Windows 子系統 (WSL) 等工具。
- 收集檔案以使用遊戲伺服器建置進行部署。在本機電腦上，建立工作目錄來組織檔案，這些檔案會內建在您的遊戲伺服器容器映像中。這些可能包括遊戲相依性、啟動遊戲伺服器的指令碼，以及啟動容器時的其他程序等。
- 新增遊戲伺服器建置的 OpenSSL 程式庫檔案。如果您的遊戲伺服器與伺服器 SDK 5.3 或更新版本整合，您可以略過此步驟。

手動尋找 OpenSSL 程式庫並將其複製到位於的遊戲組建套件目錄<YourGame>/Binaries/Win64。您必須使用與 Unreal Engine 5 版本相同的 OpenSSL 版本。使用錯誤的 OpenSSL 程式庫部署的遊戲組建將無法與服務通訊Amazon GameLift Servers。

若要尋找 OpenSSL 程式庫，請查看您的遊戲引擎來源。位置會根據您的開發環境而有所不同：

在 Windows 上：

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

在 Linux 上：

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

- 將您的遊戲用戶端程式碼與 整合Amazon GameLift Servers。完成此任務的一個方法是新增已整合的範例資產（隨附於外掛程式）。如需準備遊戲用戶端程式碼的指引，請參閱 [整合您的用戶端遊戲地圖](#)。
- 在本機電腦上安裝 Docker。如果您希望外掛程式為您建立容器映像並將其推送至 ECR 儲存庫，則需要安裝此工具。或者，您可以手動執行這些任務，並指示外掛程式使用現有的容器映像。如需手動建置映像的詳細資訊，請參閱[建置的容器映像Amazon GameLift Servers](#)。

若要啟動Amazon GameLift Servers受管容器工作流程：

- 在 Unreal 編輯器主工具列中，選擇Amazon GameLift Servers選單，然後選取受管容器。此動作會開啟外掛程式頁面 Host with Managed Containers，其中提供使用遊戲伺服器建置建立容器映像、將其部署至容器機群，以及啟動遊戲step-by-step程序。

步驟 0：設定您的設定檔

本節顯示您目前選取的使用者設定檔。確認目前的使用者設定檔是您要用於此工作流程的設定檔。您在此工作流程中建立的所有資源都與設定檔 AWS 的帳戶相關聯，並放置在設定檔的預設 AWS 區域中。設定檔使用者的許可決定您對 AWS 資源和動作的存取。

您可能需要修改選取的使用者設定檔，如果：

- 目前未選取設定檔。
- 您想要選取不同的設定檔或建立新的設定檔。
- 您需要引導選取的設定檔（如果引導狀態為非作用中）。

設定或變更選取的使用者設定檔

- 在Amazon GameLift Servers功能表中，選擇開放 AWS 存取登入資料。

步驟 1：評估容器準備程度

在將遊戲伺服器部署到容器機群之前，您必須將其封裝到容器映像中，並存放在 Amazon ECR 儲存庫中。外掛程式可以為您完成這些任務，或者您可以手動執行這些任務。在此步驟中，提供容器映像和 ECR 儲存庫狀態的相關資訊。

使用評估問題來告知外掛程式需要採取哪些步驟：

- 建立新的容器映像。如果您選擇此選項，下一步將提示您輸入遊戲伺服器建置目錄的位置和建置可執行檔。外掛程式使用 Dockerfile 範本（由提供Amazon GameLift Servers），並自動為您的遊戲進行設定。您可以在 [中檢視範本建置的容器映像 Amazon GameLift Servers](#)。選擇此選項後，請指出您希望外掛程式存放新映像的位置：
- 建立新的 Amazon ECR 儲存庫，並將容器映像推送到其中。外掛程式會使用 AWS 帳戶建立私有 ECR 儲存庫，並在您選取的使用者設定檔 AWS 區域 中建立預設值。
- 將容器映像推送至先前建立的 Amazon ECR 儲存庫。如果您選擇此選項，下一個步驟將提示您從清單中選擇現有的 Amazon ECR 儲存庫。此清單包含 AWS 帳戶的所有 Amazon ECR 儲存庫，以及所選使用者設定檔 AWS 區域 中的預設值。您可以選取公有或私有儲存庫。
- 使用現有的容器映像。如果您已手動建置映像，建議您使用提供的 Dockerfile 範本Amazon GameLift Servers，該範本可在取得[建置的容器映像 Amazon GameLift Servers](#)。選擇此選項後，請指出影像所在的位置。
- 本機存放的 Docker 產生的映像。如果您選擇此選項，外掛程式會建立新的 Amazon ECR 私有儲存庫，並將本機映像檔案推送到該儲存庫。下一個步驟會提示您輸入映像 ID，外掛程式會使用此 ID 來尋找映像檔案。
- 已存放在 Amazon ECR 儲存庫中的容器映像。如果您選擇此選項，下一個步驟將提示您從清單中選擇現有的 Amazon ECR 儲存庫和映像。此清單包含 AWS 帳戶的所有 Amazon ECR 儲存庫，以及所選使用者設定檔 AWS 區域 中的預設值。您可以選取公有或私有儲存庫。

步驟 2：設定映像部署

在此步驟中，請提供外掛程式將容器映像部署至容器機群所需的資訊。此步驟會請求以下資訊：

- 遊戲伺服器建置、容器映像或 Amazon ECR 儲存庫的位置，取決於您在步驟 1 中的選擇。
- 用於受管容器部署的案例。
- 用戶端組態輸出路徑。選取用戶端建置中包含您 AWS 組態的資料夾。在下列位置尋找它：`[client-build]/[project-name]/Content/CloudFormation`。
- 選用的部署設定。本節具有外掛程式預設使用的組態設定。您可以修改這些值或保留預設值
 - 遊戲名稱預設為遊戲專案的名稱。外掛程式建立的所有 AWS 資源都會參考遊戲名稱值。
 - 連接埠範圍、記憶體限制和 vCPU 限制是容器機群的組態設定。如需自訂這些值的詳細資訊，請參閱 [設定網路連線](#) 以取得連線連接埠範圍，以及 [設定資源限制](#) 以取得資源限制。
 - 容器映像標籤用於分類 Amazon ECR 中的容器映像。預設值為 `unreal-gamelift-plugin`。
 - Amazon ECR 儲存庫的名稱。您可以編輯此欄位，只在外掛程式為您建立 ECR 儲存庫時建議自訂名稱。預設值為 `unreal-game lift-plugin-ecr-repository`。

部署案例選項

單一區域容器機群

此案例會將您的遊戲伺服器部署到單一容器機群。這是測試與 AWS 和容器組態之伺服器整合的良好起點。它會部署下列資源。

- Amazon GameLift Servers 容器群組定義說明如何在容器機群上部署和執行容器映像。
- Amazon GameLift Servers 容器機群（隨需），您的遊戲伺服器容器已安裝並執行，並具有別名。
- Amazon Cognito 使用者集區和用戶端可讓玩家驗證和啟動遊戲。
- 將使用者集區與 API 連結APIs Gateway 授權方。
- Web 存取控制清單 (ACL)，用於調節對 API Gateway 的過多玩家呼叫。
- 代表遊戲用戶端向Amazon GameLift Servers服務提出請求的後端服務，例如請求遊戲工作階段和加入遊戲：
 - API Gateway + Lambda 函數，讓玩家請求遊戲工作階段槽。CreateGameSession() 如果沒有可用的開放插槽，此函數會呼叫。
 - API Gateway + Lambda 函數可讓玩家取得其遊戲請求的連線資訊。

使用 的單一區域容器機群 FlexMatch

此案例會將您的遊戲伺服器部署到容器機群、設定遊戲工作階段置放，以及設定FlexMatch配對。當您準備好開始為託管解決方案設計自訂配對建構器時，此案例非常有用。使用此案例來建立此解決方案的基本資源，您可以視需要稍後進行自訂。它會部署下列資源：

- Amazon GameLift Servers 容器群組定義，說明如何在容器機群上部署和執行容器映像。
- Amazon GameLift Servers 容器機群（隨需），您的遊戲伺服器容器已安裝並執行，並具有別名。
- FlexMatch 配對組態和配對規則集，以接受玩家請求並形成配對。
- Amazon GameLift Servers 透過尋找最佳的託管資源（根據可行性、成本、玩家延遲等）並啟動遊戲工作階段，滿足對提議配對的請求的遊戲工作階段佇列。
- Amazon Cognito 使用者集區和用戶端可讓玩家驗證和啟動遊戲。
- 將使用者集區與 API 連結APIs Gateway 授權方。
- Web 存取控制清單 (ACL)，用於調節對 API Gateway 的過多玩家呼叫。
- 代表遊戲用戶端向Amazon GameLift Servers服務提出請求的後端服務，例如請求遊戲工作階段和加入遊戲：

- API Gateway + Lambda 函數，讓玩家請求遊戲工作階段槽。StartMatchmaking() 如果沒有可用的開放插槽，此函數會呼叫。
- API Gateway + Lambda 函數可讓玩家取得其遊戲請求的連線資訊。
- DynamoDB 資料表可存放玩家的配對票證和遊戲工作階段資訊。
- Amazon SNS 主題 + Lambda 函數來處理 GameSessionQueue 事件。

部署容器機群

機群組態完成後，請選擇部署容器機群按鈕以開始部署。當外掛程式建立容器映像並將其推送至 ECR、為容器機群佈建託管資源、為選取的託管解決方案案例部署機群和其他 AWS 資源時，此程序可能需要幾分鐘的時間。

開始部署時，您可以追蹤每個步驟的進度。根據您的組態，這些步驟可能包含下列項目：

- 設定容器映像
- 建立 Amazon ECR 儲存庫
- 建置映像並推送至 Amazon ECR
- 建立容器群組定義
- 建立容器機群

如需更詳細的部署資訊，請選擇 管理主控台 中的 AWS 檢視。當容器機群達到作用中狀態時，機群會主動執行具有伺服器程序的容器，這些程序已準備好託管遊戲工作階段。

部署完成時，您會有一個工作容器機群，已準備好託管遊戲工作階段並接受玩家連線。

您無法停止進行中的部署。如果部署進入錯誤狀態或失敗，您可以使用重設部署選項重新開始。

啟動用戶端

此時，您已完成所有任務，以啟動和播放使用託管的多玩家遊戲 Amazon GameLift Servers。若要玩遊戲，請選擇啟動用戶端以啟動遊戲用戶端的本機執行個體。

- 如果您部署了單一機群案例，請使用一個玩家開啟遊戲用戶端的一個執行個體，然後輸入伺服器映射以四處移動。您可以開啟遊戲用戶端的第二個執行個體，將第二個玩家新增至相同的伺服器遊戲地圖。

- 如果您部署了FlexMatch案例，託管解決方案會等待至少兩個遊戲用戶端提出配對請求。使用一個玩家開啟遊戲用戶端的至少兩個執行個體。這兩位玩家將會進行配對，並提示加入配對的遊戲工作階段。

更新容器機群

如果您已成功部署受管容器託管解決方案，您可以使用更新部署功能。此選項可讓您更新已部署容器機群的組態設定，而不必建立新的機群。

更新部署時，您可以使用不同的遊戲伺服器建置來部署容器映像、變更 Amazon ECR 儲存庫、選擇不同的部署案例，以及自訂選用的組態設定。

當您準備好部署變更時，請選擇更新。部署更新所需的時間類似於完整部署。如需詳細部署資訊，請選擇 AWS 管理主控台中的檢視。

清除已部署的資源

最佳實務是，當您不再需要受管容器解決方案時，請立即清理這些解決方案 AWS 的資源。如果您不移除這些資源，則可能會繼續產生這些資源的成本。

刪除下列資源：

- 受管容器資源堆疊。此堆疊中的資源取決於您選擇的部署案例。若要刪除整個堆疊，請使用 AWS CloudFormation 主控台。從Amazon GameLift Servers外掛程式產生的堆疊使用以下命名慣例：GameLiftPluginForUnreal-{GameName}-Containers。在外掛程式中啟動新的受管容器部署之前，請等待堆疊刪除程序完成。如需詳細資訊，請參閱[從 CloudFormation 主控台刪除堆疊](#)。
- Amazon ECR 儲存庫。如果您使用外掛程式為容器映像建立儲存庫，您可能想要刪除任何不再需要的儲存庫。在重設受管容器部署之前，您不需要刪除儲存庫。如果您更新或重設部署，外掛程式將自動使用相同的儲存庫，除非被指示使用另一個儲存庫。如需詳細資訊，請參閱[刪除 Amazon ECR 中的私有儲存庫](#)。

Amazon GameLift ServersUnity 的外掛程式 (伺服器 SDK 5.x)

此外掛程式會將 Amazon GameLift Servers C# 伺服器 SDK 和工具新增至 Unity 編輯器。使用引導式 UI 工作流程將伺服器 SDK 功能整合到您的遊戲專案中，並為您的遊戲伺服器部署Amazon GameLift Servers託管解決方案。

使用 外掛程式，您可以建置基本的工作託管解決方案，然後視需要最佳化和自訂。使用本機工作站做為主機來設定 Amazon GameLift Servers Anywhere 機群。對於使用受管 EC2 或受管容器機群的雲端託管，請使用完整的解決方案部署遊戲伺服器，以管理遊戲工作階段請求和用戶端連線。

主題

- [安裝 Unity 遊戲專案的外掛程式](#)
- [Unity 的外掛程式：設定 AWS 使用者設定檔](#)
- [Unity 的外掛程式：使用 Amazon GameLift ServersAnywhere 設定本機測試](#)
- [Unity 外掛程式：將您的遊戲部署到受管 EC2 機群](#)
- [Unity 外掛程式：將您的遊戲部署到受管容器機群](#)

安裝 Unity 遊戲專案的外掛程式

[從 GitHub 取得 Unity 的 Amazon GameLift Servers 外掛程式](#)

如需如何安裝遊戲專案外掛程式的詳細資訊，請參閱 GitHub 儲存庫讀我檔案。

外掛程式包含下列元件：

- Unity 編輯器的外掛程式模組。安裝外掛程式時，新的主功能表項目可讓您存取 Amazon GameLift Servers 功能。
- 具有用戶端功能的 Amazon GameLift Servers 服務 API 的 C# 程式庫。
- Amazon GameLift Servers 伺服器 SDK (5.x 版) 的 C# 程式庫。
- 遊戲內容範例，包括資產和場景，即使您沒有可供建置的多玩家遊戲 Amazon GameLift Servers，也可以試用。
- 作為 AWS CloudFormation 範本提供的解決方案組態，外掛程式會在將您的遊戲伺服器部署到雲端進行託管時使用。

此外掛程式使用 AWS CloudFormation 範本來部署常見遊戲案例的託管解決方案。依照提供的方式使用這些解決方案，或視需要為您的遊戲自訂解決方案。

Unity 的外掛程式：設定 AWS 使用者設定檔

安裝外掛程式後，請使用有效的 AWS 帳戶設定使用者設定檔。您可以在外掛程式中維護多個設定檔，但一次只能選取一個設定檔。每當您在外掛程式中工作時，請選取要使用的設定檔。每個工作流程頁面會顯示目前選取的設定檔。

維護多個設定檔可讓您在不同託管部署之間切換。例如，您可以設定使用相同 AWS 帳戶但部署到不同 AWS 區域的設定檔。或者，您可以使用不同的 AWS 帳戶或使用者和許可集來設定設定檔。

Note

如果您已在工作站上安裝 AWS CLI 並已設定設定檔，Amazon GameLift Servers 外掛程式會偵測它並將其列為現有的設定檔。外掛程式會自動選取任何名為 `default` 的設定檔。您可以使用現有的設定檔或建立新的設定檔。

必須引導所有設定檔，才能在您的帳戶使用者下設定一些必要的 AWS 資源。

管理您的 AWS 設定檔

1. 在 Unity 主工具列中，選擇 Amazon GameLift Servers 選單，然後選取 AWS 存取登入資料。此動作會將 Amazon GameLift Servers 外掛程式開啟至 頁面設定您的使用者設定檔。
2. 使用按鈕來建立新 AWS 帳戶，或為您已有 AWS 的帳戶設定使用者設定檔。
3. 如果您還沒有使用者設定檔，系統會提示您輸入設定檔詳細資訊並建立新的設定檔。請提供下列資訊：
 - AWS 帳戶。如果您建立新 AWS 帳戶，請使用 的連結 AWS Management Console 並依照提示操作。如需詳細資訊，[請參閱建立 AWS 帳戶](#)。
 - 具有使用 Amazon GameLift Servers 和其他必要 AWS 服務許可 AWS 的使用者。[設定 AWS 帳戶](#) 如需設定具有 Amazon GameLift Servers 許可的 AWS Identity and Access Management (IAM) 使用者，以及使用長期憑證進行程式設計存取的指示，請參閱。
 - 您 AWS 使用者的登入資料。這些登入資料包含 AWS 存取金鑰 ID 和 AWS 私密金鑰。如需詳細資訊，請參閱[取得您的存取金鑰](#)。
 - AWS 區域。這是您要建立 AWS 資源以進行託管的地理位置。在開發期間，我們建議您使用靠近實體位置的區域。從[支援的區域清單中選擇 AWS 區域](#)。
4. 如果外掛程式偵測到現有的設定檔，則會顯示可用的設定檔清單。從清單中選擇現有的設定檔，或選擇新增另一個設定檔以建立新的設定檔。

引導使用者設定檔

所有設定檔都必須引導才能與 Amazon GameLift Servers 外掛程式搭配使用。引導會建立設定檔專屬的 Amazon S3 儲存貯體。它用於存放專案組態、建置成品和其他相依性。儲存貯體不會在其他設定檔之間共用。

引導涉及建立新的 AWS 資源，並可能產生成本。

若要引導您的設定檔：

1. 在AWS 存取登入資料頁面上，檢查您要使用的使用者設定檔的引導狀態。如果設定檔的引導狀態為「非作用中」且未列出 S3 儲存貯體，則需要引導設定檔。
2. 選取您要使用的設定檔，然後選擇引導設定檔。
3. 等待引導狀態變更為「作用中」。這可能需要幾分鐘的時間。

Unity 的外掛程式：使用 Amazon GameLift ServersAnywhere 設定本機測試

在此工作流程中，您可以新增 Amazon GameLift Servers 功能的用戶端和伺服器遊戲程式碼，並使用外掛程式將本機工作站指定為測試遊戲伺服器主機。當您完成整合任務時，請使用 外掛程式來建置遊戲用戶端和伺服器元件。

若要啟動 Amazon GameLift Servers Anywhere 工作流程：

- 在 Unity 編輯器主功能表中，選擇Amazon GameLift Servers並選取 Host with Anywhere。此動作會開啟外掛程式頁面，以使用 @Anywhere 機群設定您的遊戲。此頁面提供五個步驟的程序來整合、建置和啟動遊戲元件。

設定您的設定檔

選擇您要在遵循此工作流程時使用的設定檔。您選取的設定檔會影響工作流程中的所有步驟。您建立的所有資源都會與設定檔 AWS 的帳戶相關聯，並放置在設定檔的預設 AWS 區域中。設定檔使用者的許可決定您對 AWS 資源和動作的存取。

1. 從可用設定檔的下拉式清單中選取設定檔。如果您還沒有設定檔，或想要建立新的設定檔，請前往Amazon GameLift Servers選單，然後選擇設定 AWS 帳戶設定檔。
2. 如果引導狀態不是「作用中」，請選擇引導設定檔，並等待狀態變更為「作用中」。

將您的遊戲程式碼與 C# 伺服器 SDK 整合

Note

如果您匯入了範例遊戲，則可以略過此步驟。範例遊戲資產已有必要的伺服器和用戶端程式碼。

對於工作流程中的此步驟，您會更新遊戲專案中的用戶端和伺服器程式碼。

- * 遊戲伺服器必須能夠與服務通訊Amazon GameLift Servers，以接收啟動遊戲工作階段、提供遊戲工作階段連線資訊以及報告狀態的提示。
- 遊戲用戶端必須能夠取得遊戲工作階段的相關資訊、加入或啟動遊戲工作階段，以及取得連線資訊以加入遊戲。

整合您的伺服器程式碼

如果您使用自己的遊戲專案搭配自訂場景，請使用提供的範例程式碼將必要的伺服器程式碼新增至您的遊戲專案：

1. 在您的遊戲專案檔案中，開啟 Assets/Scripts/Server 資料夾。如果不存在，請建立它。
2. 前往 GitHub repo [aws/amazon-gamelift-plugin-unity](https://github.com/aws/amazon-gamelift-plugin-unity) 並開啟路徑 Samples~/SampleGame/Assets/Scripts/Server。
3. 找到 檔案並將其GameLiftServer.cs複製到遊戲專案的 Server 資料夾。當您建置伺服器可執行檔時，請使用此檔案做為建置目標。

範例程式碼包含這些最低必要元素，這些元素使用 Amazon GameLift Servers C# 伺服器 SDK（第 5 版）：

- 初始化 Amazon GameLift Servers API 用戶端。Amazon GameLift Servers Anywhere 機群需要具有伺服器參數的InitSDK()呼叫。這些設定會自動設定為在 外掛程式中使用。
- 實作必要的回呼函數，以回應來自 Amazon GameLift Servers服務的請求，包括 OnStartGameSession、OnProcessTerminate和 onHealthCheck。
- ProcessReady() 使用指定的連接埠呼叫，以在伺服器程序準備好託管遊戲工作階段時通知 Amazon GameLift Servers服務。

如果您想要自訂範例伺服器程式碼，請參閱這些資源：

- [Amazon GameLift Servers 新增至您的遊戲伺服器](#)
- [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)

整合您的用戶端程式碼

如果您使用自己的遊戲專案搭配自訂場景，則需要將基本功能整合到您的遊戲用戶端。您也需要新增 UI 元素，讓玩家可以登入並加入遊戲工作階段。使用適用於的服務 API Amazon GameLift Servers (在 AWS SDK 中) 取得遊戲工作階段資訊、建立新的遊戲工作階段，或加入現有的遊戲工作階段，

使用 Anywhere 機群建置用戶端進行本機測試時，您可以將直接呼叫新增至 Amazon GameLift Servers 服務。當您為雲端託管開發遊戲時，或者如果您計劃使用 Anywhere 機群進行生產託管時，您將需要建立用戶端後端服務，以處理遊戲用戶端與 Amazon GameLift Servers 服務之間的所有通訊。

若要 Amazon GameLift Servers 整合到您的用戶端程式碼，請使用下列資源做為指南。

- 將用戶端與 GitHub repo `aws/amazon-gamelift-plugin-unity` 中的 `GameLiftCoreApi` 類別整合。此類別提供玩家身分驗證和擷取遊戲工作階段資訊的控制項。
- 檢視遊戲整合範例，可在 GitHub repo `aws/amazon-gamelift-plugin-unity` 中取得 `Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`。
- 請遵循將 Amazon GameLift Servers 新增至 Unity 遊戲用戶端中的指示。

對於連接到 Anywhere 機群的遊戲用戶端，您的遊戲用戶端需要以下資訊。外掛程式會自動更新您的遊戲專案，以使用您在外掛程式中建立的資源。

- `FleetId` - Anywhere 機群的唯一識別符。
- `FleetLocation` - Anywhere 機群的自訂位置。
- `AwsRegion` - Anywhere 機群託管 AWS 的區域。這是您在使用者設定檔中設定的區域。
- `ProfileName` - 本機電腦上的 AWS 登入資料設定檔，允許存取適用於 Amazon GameLift Servers 的 AWS SDK。遊戲用戶端使用這些登入資料來驗證對 Amazon GameLift Servers 服務的請求。

Note

登入資料設定檔是由外掛程式產生，並存放在本機電腦上。因此，您必須在本機機器（或具有相同設定檔的機器）上執行用戶端。

連線至 Anywhere 機群

在此步驟中，您會指定要使用的 Anywhere 機群。Anywhere 機群會定義一組運算資源，這些資源可以位於任何地方，用於遊戲伺服器託管。

- 如果您目前正在使用的 AWS 帳戶有現有的 Anywhere 機群，請開啟機群名稱下拉式清單欄位，然後選擇機群。此下拉式清單只會顯示目前作用中使用者設定檔 AWS 區域中的 Anywhere 機群。
- 如果沒有現有機群，或者您想要建立新的機群，請選擇建立新的 Anywhere 機群並提供機群名稱。

在您為專案選擇 Anywhere 機群之後，Amazon GameLift Servers 會驗證機群狀態為作用中，並顯示機群 ID。您可以在 Unity 編輯器的輸出日誌中追蹤此請求的進度。

註冊運算

在此步驟中，您會將本機工作站註冊為新 Anywhere 機群中的運算資源。

1. 輸入本機電腦的運算名稱。如果您在機群中新增多個運算，名稱必須是唯一的。
2. 選擇註冊運算。您可以在 Unity 編輯器的輸出日誌中追蹤此請求的進度。

外掛程式會使用設定為 localhost (127.0.0.1) 的 IP 地址註冊您的本機工作站。此設定假設您將在相同的機器上執行遊戲用戶端和伺服器。

為了回應此動作，Amazon GameLift Servers 會驗證它是否可以連接到運算，並傳回有關新註冊的運算的資訊。

啟動遊戲

在此步驟中，您將建置遊戲元件並啟動它們來玩遊戲。完成下列任務：

1. 設定您的遊戲用戶端。在此步驟中，您會提示外掛程式更新遊戲專案的GameLiftClientSettings資產。外掛程式使用此資產來存放遊戲用戶端連線至Amazon GameLift Servers服務所需的特定資訊。
 - a. 如果您未匯入並初始化範例遊戲，請建立新的GameLiftClientSettings資產。在 Unity 編輯器主功能表中，選擇資產、建立、Amazon GameLift、用戶端設定。如果您在專案GameLiftClientSettings中建立多個 複本，外掛程式會自動偵測此情況，並通知您外掛程式將更新的資產。
 - b. 在啟動遊戲中，選擇設定用戶端：Application Anywhere 設定。此動作會更新您的遊戲用戶端設定，以使用您剛設定的 Anywhere 機群。

2. 建置並執行您的遊戲用戶端。

- a. 使用標準 Unity 建置程序建置用戶端可執行檔。在檔案、建置設定中，將平台切換到 Windows、Mac、Linux。如果您匯入範例遊戲並初始化設定，則建置清單和建置目標會自動更新。
 - b. 啟動新建置遊戲用戶端可執行檔的一或多個執行個體。
3. 在 Anywhere 機群中啟動遊戲伺服器。選擇伺服器：在編輯器中啟動伺服器。只要 Unity 編輯器保持開啟狀態，此任務就會啟動用戶端可以連線的即時伺服器。
 4. 開始或加入遊戲工作階段。在您的遊戲用戶端執行個體中，使用 UI 將每個用戶端加入遊戲工作階段。如何執行此操作取決於如何將功能新增至用戶端。

如果您使用的是範例遊戲用戶端，則具有下列特性：

- 玩家登入元件。連線至 Anywhere 機群上的遊戲伺服器時，沒有玩家驗證。您可以輸入任何值來加入遊戲工作階段。
- 簡單聯結遊戲 UI。當用戶端嘗試加入遊戲時，用戶端會自動尋找具有可用玩家位置的作用中遊戲工作階段。如果沒有可用的遊戲工作階段，用戶端會請求新的遊戲工作階段。如果遊戲工作階段可用，用戶端會請求加入可用的遊戲工作階段。使用多個並行用戶端測試遊戲時，第一個用戶端會啟動遊戲工作階段，其餘用戶端會自動加入現有的遊戲工作階段。
- 具有四個玩家位置的遊戲工作階段。您可以同時啟動最多四個遊戲用戶端執行個體，它們將加入相同的遊戲工作階段。

從伺服器可執行檔啟動（選用）

您可以建置和啟動遊戲伺服器可執行檔，以便在 Anywhere 機群上進行測試。

1. 使用標準 Unity 建置程序建置伺服器可執行檔。在檔案、建置設定中，將平台切換至專用伺服器和建置。
2. 使用 Anywhere 機群 ID 和 AWS 區域呼叫 AWS CLI 命令 [get-compute-auth-token](#)，以取得短期身分驗證字符。當您建立機群時，機群 ID 會顯示在連線至 Anywhere 機群中。當您選取作用中的設定檔時，AWS 區域會顯示在設定設定檔中。

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

3. 從命令列啟動新建置的遊戲伺服器可執行檔，並傳遞有效的身分驗證字符。

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

Unity 外掛程式：將您的遊戲部署到受管 EC2 機群

在此工作流程中，您會使用 外掛程式來準備您的遊戲，以便在由 管理的雲端型運算資源上託管 Amazon GameLift Servers。您可以為 Amazon GameLift Servers 功能新增用戶端和伺服器遊戲程式碼，然後將伺服器組建上傳到 Amazon GameLift Servers 服務以進行託管。當此工作流程完成時，您會在雲端中執行遊戲伺服器，以及可以與其連線的工作遊戲用戶端。

若要啟動 Amazon GameLift Servers 受管 Amazon EC2 工作流程：

- 在 Unity 編輯器主功能表中，選擇 Amazon GameLift Servers 並選取 Host with Managed EC2。此工作流程提供六個步驟來整合、建置、部署和啟動遊戲元件。

設定您的設定檔

選擇您要在遵循此工作流程時使用的設定檔。您選取的設定檔會影響工作流程中的所有步驟。您建立的所有資源都會與設定檔 AWS 的帳戶相關聯，並放置在設定檔的預設 AWS 區域中。設定檔使用者的許可決定您對 AWS 資源和動作的存取。

- 從可用設定檔的下拉式清單中選取設定檔。如果您還沒有設定檔，或想要建立新的設定檔，請前往 Amazon GameLift Servers 選單並選擇設定 AWS 帳戶設定檔。
- 如果引導狀態不是「作用中」，請選擇引導設定檔，並等待狀態變更為「作用中」。

將您的遊戲與 整合 Amazon GameLift Servers

在此任務中，您會更新遊戲專案中的用戶端和伺服器程式碼。

- 遊戲伺服器必須能夠與服務通訊 Amazon GameLift Servers，以接收啟動遊戲工作階段、提供遊戲工作階段連線資訊以及報告狀態的提示。
- 遊戲用戶端必須能夠取得遊戲工作階段的相關資訊、加入或啟動遊戲工作階段，以及取得連線資訊以加入遊戲。

 Note

如果您匯入了範例遊戲，則可以略過此步驟。範例遊戲資產已有必要的伺服器 and 用戶端程式碼。

整合您的伺服器程式碼

使用您自己的遊戲專案搭配自訂場景時，請使用提供的範例程式碼，將必要的伺服器程式碼新增至您的遊戲專案。如果您將遊戲專案與 Anywhere 機群整合進行測試，表示您已完成此步驟中的指示。

1. 在您的遊戲專案檔案中，開啟 Assets/Scripts/Server 資料夾。如果不存在，請建立它。
2. 前往 GitHub repo [aws/amazon-gamelift-plugin-unity](https://github.com/aws/amazon-gamelift-plugin-unity) 並開啟路徑 Samples~/SampleGame/Assets/Scripts/Server。
3. 找到 檔案並將其GameLiftServer.cs複製到遊戲專案的 Server 資料夾。當您建置伺服器可執行檔時，請使用此檔案做為建置目標。

範例程式碼包含這些最低必要元素，這些元素使用 Amazon GameLift Servers C# 伺服器 SDK（第 5 版）：

- 初始化 Amazon GameLift Servers API 用戶端。Amazon GameLift Servers Anywhere 機群需要具有伺服器參數的 InitSDK() 呼叫。這些設定會自動設定為在 外掛程式中使用。
- 實作必要的回呼函數，以回應來自 Amazon GameLift Servers服務的請求，包括 OnStartGameSession、OnProcessTerminate和 onHealthCheck。
- ProcessReady() 使用指定的連接埠呼叫，以在伺服器程序準備好託管遊戲工作階段時通知 Amazon GameLift Servers服務。

如果您想要自訂範例伺服器程式碼，請參閱這些資源：

- [Amazon GameLift Servers 新增至您的遊戲伺服器](#)
- [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)

整合您的用戶端程式碼

對於連線至雲端遊戲伺服器的遊戲用戶端，最佳實務是使用用戶端後端服務來呼叫Amazon GameLift Servers服務，而不是直接從遊戲用戶端進行呼叫。

在託管於受管 EC2 機群的外掛程式工作流程中，每個部署案例都包含預先建置的後端服務，其中包含下列元件：

- 一組 Lambda 函數和 DynamoDB 資料表，用於請求遊戲工作階段和擷取遊戲工作階段資訊。這些元件使用 API 閘道做為代理。
- 產生唯一玩家 IDs 並驗證玩家連線的 Amazon Cognito 使用者集區。

若要使用這些元件，您的遊戲用戶端需要將請求傳送至後端服務的功能，才能執行下列動作：

- 在 AWS Cognito 使用者集區中建立玩家使用者並進行驗證。
- 加入遊戲工作階段並接收連線資訊。
- 使用配對來加入遊戲。

使用下列資源做為指南。

- 將用戶端與 GitHub repo [aws/amazon-gamelift-plugin-unity](#) 中的 [GameLiftCoreApi](#) 類別整合。此類別提供玩家身分驗證和擷取遊戲工作階段資訊的控制項。
- 若要檢視範例遊戲整合，請前往 GitHub repo [aws/amazon-gamelift-plugin-unity](#)、Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs。
- [Amazon GameLift Servers 新增至您的遊戲用戶端](#)。

選取部署案例

在此步驟中，您可以選擇目前要部署的遊戲託管解決方案。您可以使用任何案例，對遊戲進行多個部署。

- **單一區域機群：**將您的遊戲伺服器部署到作用中設定檔預設 AWS 區域中的單一託管資源機群。此案例是測試與 AWS 和 伺服器建置組態之伺服器整合的良好起點。它會部署下列資源：
 - AWS 機群（隨需），您的遊戲伺服器建置已安裝並執行。
 - Amazon Cognito 使用者集區和用戶端可讓玩家驗證和啟動遊戲。
 - 將使用者集區與 API 連結 APIs 閘道授權方。
 - WebACI 用於調節對 API 閘道的過多玩家呼叫。
 - API 閘道 + Lambda 函數，讓玩家請求遊戲位置。CreateGameSession() 如果沒有可用，此函數會呼叫。
 - API 閘道 + Lambda 函數可讓玩家取得其遊戲請求的連線資訊。

- FlexMatch 機群：將您的遊戲伺服器部署到一組機群，並使用規則設定 FlexMatch 配對建構器來建立玩家配對。此案例使用具有多機群、多位置結構的低成本 Spot 託管，以實現持久的可用性。當您準備好開始為託管解決方案設計配對建構器元件時，此方法非常有用。在此案例中，您將為此解決方案建立基本資源，之後可以視需要自訂。它會部署下列資源：
 - FlexMatch 配對組態和配對規則集，以接受玩家請求並形成配對。
 - 安裝 AWS 遊戲伺服器組建並在多個位置執行的三個機群。包含兩個 Spot 機群和一個隨需機群做為備份。
 - AWS 遊戲工作階段置放佇列，透過尋找最佳的可能託管資源（根據可行性、成本、玩家延遲等）並啟動遊戲工作階段，來滿足提議配對的請求。
 - Amazon Cognito 使用者集區和用戶端可讓玩家驗證和啟動遊戲。
 - 將使用者集區與 API 連結 APIs 閘道授權方。
 - WebACI 用於調節對 API 閘道的過多玩家呼叫。
 - API 閘道 + Lambda 函數，讓玩家請求遊戲位置。此函數會呼叫 `StartMatchmaking()`。
 - API 閘道 + Lambda 函數可讓玩家取得其遊戲請求的連線資訊。
 - Amazon DynamoDB 資料表，用於存放玩家和遊戲工作階段資訊的配對票證。
 - SNS 主題 + 處理 `GameSessionQueue` 事件的 Lambda 函數。

設定遊戲參數

在此步驟中，您會描述要上傳到的遊戲 AWS。

- 遊戲名稱：為您的遊戲專案提供有意義的名稱。此名稱用於 外掛程式。
- 機群名稱：為您的受管 EC2 機群提供有意義的名稱。在主控台中列出資源時，Amazon GameLift Servers 會使用此名稱 AWS（以及機群 ID）。
- 組建名稱：為您的伺服器組建提供有意義的名稱。AWS 使用此名稱來參考上傳到 Amazon GameLift Servers 並用於部署的伺服器組建複本。
- 啟動參數：輸入在受管 EC2 機群執行個體上啟動伺服器可執行檔時要執行的選用指示。長度上限為 1024 個字元。
- 遊戲伺服器資料夾：提供包含伺服器建置的本機資料夾路徑。
- 遊戲伺服器檔案：指定伺服器可執行檔名稱。

部署案例

在此步驟中，您會根據您選擇的部署案例，將遊戲部署至雲端託管解決方案。此程序可能需要幾分鐘的時間，同時 AWS 驗證您的伺服器建置、佈建託管資源、安裝遊戲伺服器、啟動伺服器程序，以及讓他們準備好託管遊戲工作階段。

若要開始部署，請選擇部署 CloudFormation。您可以在這裡追蹤遊戲託管的狀態。如需更多詳細資訊，您可以登入的 AWS AWS 管理主控台，並檢視事件通知。請務必使用與外掛程式中作用中使用者設定檔相同的帳戶、使用者和 AWS 區域來登入。

部署完成時，您會在 AWS EC2 執行個體上安裝遊戲伺服器。至少有一個伺服器程序正在執行，並準備好啟動遊戲工作階段。

啟動遊戲用戶端

成功部署機群時，您現在已讓遊戲伺服器執行並可用於託管遊戲工作階段。您現在可以建置用戶端、啟動用戶端、連線以加入遊戲工作階段。

1. 設定您的遊戲用戶端。在此步驟中，您會提示外掛程式更新遊戲專案的GameLiftClientSettings資產。外掛程式會使用此資產來存放遊戲用戶端連線至Amazon GameLift Servers服務所需的特定資訊。
 - a. 如果您未匯入並初始化範例遊戲，請建立新的GameLiftClientSettings資產。在 Unity 編輯器主功能表中，選擇資產、建立、Amazon GameLift、用戶端設定。如果您在專案GameLiftClientSettings中建立多個複本，外掛程式會自動偵測此情況，並通知您外掛程式將更新的資產。
 - b. 在啟動遊戲中，選擇設定用戶端：套用受管 EC2 設定。此動作會更新您的遊戲用戶端設定，以使用您剛部署的受管 EC2 機群。
2. 建置您的遊戲用戶端。使用標準 Unity 建置程序建置用戶端可執行檔。在檔案、建置設定中，將平台切換至 Windows、Mac、Linux。如果您匯入範例遊戲並初始化設定，則建置清單和建置目標會自動更新。
3. 啟動新建置的遊戲用戶端可執行檔。若要開始玩遊戲，請啟動兩到四個用戶端執行個體，並使用每個執行個體中的 UI 來加入遊戲工作階段。

如果您使用的是範例遊戲用戶端，則具有下列特性：

- 玩家登入元件。連線至 Anywhere 機群上的遊戲伺服器時，沒有玩家驗證。您可以輸入任何值來加入遊戲工作階段。

- 簡單聯結遊戲 UI。當用戶端嘗試加入遊戲時，用戶端會自動尋找具有可用玩家位置的作用中遊戲工作階段。如果沒有可用的遊戲工作階段，用戶端會請求新的遊戲工作階段。如果遊戲工作階段可用，用戶端會請求加入可用的遊戲工作階段。使用多個並行用戶端測試遊戲時，第一個用戶端會啟動遊戲工作階段，其餘用戶端會自動加入現有的遊戲工作階段。
- 具有四個玩家位置的遊戲工作階段。您可以同時啟動最多四個遊戲用戶端執行個體，它們將加入相同的遊戲工作階段。

Unity 外掛程式：將您的遊戲部署到受管容器機群

使用此引導式外掛程式工作流程為您的遊戲伺服器建立容器映像，並將其部署到容器型託管解決方案。當您成功完成此工作流程時，您的容器化遊戲伺服器會在雲端執行，而且您可以使用 外掛程式來啟動遊戲用戶端、連線至遊戲工作階段，以及玩遊戲。

開始之前

此工作流程假設您已完成下列任務。

- 將您的遊戲伺服器程式碼與Amazon GameLift Servers伺服器 SDK 整合。我們的託管遊戲伺服器必須能夠與 Amazon GameLift Servers服務通訊，以便回應啟動新遊戲工作階段和報告遊戲工作階段狀態的請求。如果您尚未完成此任務，建議您先使用 Anywhere 遵循外掛程式工作流程主機。如需準備遊戲伺服器程式碼的指引，請參閱 [整合您的伺服器程式碼](#)。對於受管容器機群，您必須將遊戲與伺服器 SDK 5.2 版或更新版本整合。

Note

如果您使用的是啟動遊戲地圖，則此任務已為您完成。

- 封裝遊戲伺服器可執行檔以在 Linux 上執行。
- 收集檔案以使用遊戲伺服器建置進行部署。在本機電腦上，建立工作目錄來組織檔案，這些檔案將內建在您的遊戲伺服器容器映像中。這些可能包括遊戲相依性、啟動遊戲伺服器的指令碼，以及啟動容器時的其他程序等。
- 將您的遊戲用戶端程式碼與 整合Amazon GameLift Servers。完成此任務的一個方法是新增已整合的範例資產（隨附於外掛程式）。如需準備遊戲用戶端程式碼的指引，請參閱 [整合您的用戶端程式碼](#)。
- 在本機電腦上安裝 Docker。如果您希望外掛程式為您建立容器映像並將其推送至 ECR 儲存庫，則需要安裝此工具。或者，您可以手動執行這些任務，並指示外掛程式使用現有的容器映像。如需手動建置映像的詳細資訊，請參閱[建置的容器映像Amazon GameLift Servers](#)。

若要啟動Amazon GameLift Servers受管容器工作流程：

- 在 Unity 編輯器主工具列中，選擇Amazon GameLift Servers選單，然後選取受管容器。此動作會開啟外掛程式頁面 Host with Managed Containers，其中提供使用遊戲伺服器建置建立容器映像、將其部署到容器機群，以及啟動遊戲step-by-step程序。

步驟 0：設定您的設定檔

本節顯示您目前選取的使用者設定檔。確認目前的使用者設定檔是您要用於此工作流程的設定檔。您在此工作流程中建立的所有資源都與設定檔 AWS 的帳戶相關聯，並放置在設定檔的預設 AWS 區域中。設定檔使用者的許可決定您對 AWS 資源和動作的存取。

如果符合下列條件，您可能需要修改選取的使用者設定檔：

- 目前未選取任何設定檔。
- 您想要選取不同的設定檔或建立新的設定檔。
- 您需要引導選取的設定檔（如果引導狀態為非作用中）。

設定或變更選取的使用者設定檔

- 在Amazon GameLift Servers功能表中，選擇開放 AWS 存取登入資料。

步驟 1：評估容器準備程度

在將遊戲伺服器部署到容器機群之前，您必須將其封裝到容器映像中，並存放在 Amazon ECR 儲存庫中。外掛程式可以為您完成這些任務，或者您可以手動執行這些任務。在此步驟中，提供容器映像和 ECR 儲存庫狀態的相關資訊。

使用評估問題來告知外掛程式需要採取哪些步驟：

- 建立新的容器映像。如果您選擇此選項，下一步將提示您輸入遊戲伺服器建置目錄的位置和建置可執行檔。外掛程式使用 Dockerfile 範本（由提供Amazon GameLift Servers），並自動為您的遊戲進行設定。您可以在中檢視範本[建置的容器映像 Amazon GameLift Servers](#)。選擇此選項後，請指出您希望外掛程式存放新映像的位置：
- 建立新的 Amazon ECR 儲存庫，並將容器映像推送到其中。外掛程式會使用 AWS 帳戶建立私有 ECR 儲存庫，並在您選取的使用者設定檔 AWS 區域中建立預設值。

- 將容器映像推送至先前建立的 Amazon ECR 儲存庫。如果您選擇此選項，下一個步驟將提示您從清單中選擇現有的 Amazon ECR 儲存庫。此清單包含 AWS 帳戶的所有 Amazon ECR 儲存庫，以及所選使用者設定檔 AWS 區域 中的預設值。您可以選取公有或私有儲存庫。
- 使用現有的容器映像。如果您已手動建置映像，建議您使用 提供的 Dockerfile 範本 Amazon GameLift Servers，該範本可在 取得 [建置 的容器映像 Amazon GameLift Servers](#)。選擇此選項後，請指出影像所在的位置。
- 本機存放的 Docker 產生的映像。如果您選擇此選項，外掛程式會建立新的 Amazon ECR 私有儲存庫，並將本機映像檔案推送到該儲存庫。下一個步驟會提示您輸入影像 ID，外掛程式會使用此 ID 來尋找影像檔案。
- 已存放在 Amazon ECR 儲存庫中的容器映像。如果您選擇此選項，下一步將提示您從清單中選擇現有的 Amazon ECR 儲存庫和映像。此清單包含 AWS 帳戶的所有 Amazon ECR 儲存庫，以及所選使用者設定檔 AWS 區域 中的預設值。您可以選取公有或私有儲存庫。

步驟 2：設定映像部署

在此步驟中，請提供外掛程式將容器映像部署至容器機群所需的資訊。此步驟會請求以下資訊：

- 遊戲伺服器建置、容器映像或 Amazon ECR 儲存庫的位置，取決於您在步驟 1 中的選擇。
- 用於受管容器部署的案例。
- 選用部署設定。本節具有外掛程式預設使用的組態設定。您可以修改這些值或保留預設值
 - 遊戲名稱預設為遊戲專案的名稱。外掛程式建立的所有 AWS 資源都會參考遊戲名稱值。
 - 連接埠範圍、記憶體限制和 vCPU 限制是容器機群的組態設定。如需自訂這些值的詳細資訊，請參閱 [設定網路連線](#) 以取得連線連接埠範圍，以及 [設定資源限制](#) 以取得資源限制。
 - 容器映像標籤用於分類 Amazon ECR 中的容器映像。預設值為 unity-gamelift-plugin。

部署案例選項

單一區域容器機群

此案例會將您的遊戲伺服器部署到單一容器機群。這是測試與 AWS 和容器組態之伺服器整合的良好起點。它會部署下列資源。

- Amazon GameLift Servers 容器群組定義說明如何在容器機群上部署和執行容器映像。
- Amazon GameLift Servers 安裝並執行遊戲伺服器容器的容器機群（隨需），以及別名。
- Amazon Cognito 使用者集區和用戶端可讓玩家驗證和啟動遊戲。

- 將使用者集區與 API 連結APIs Gateway 授權方。
- Web 存取控制清單 (ACL)，用於調節對 API Gateway 的過多玩家呼叫。
- 代表遊戲用戶端向Amazon GameLift Servers服務提出請求的後端服務，例如請求遊戲工作階段和加入遊戲：
 - API Gateway + Lambda 函數，讓玩家請求遊戲工作階段槽。CreateGameSession() 如果沒有可用的開放插槽，此函數會呼叫。
 - API Gateway + Lambda 函數，讓玩家取得其遊戲請求的連線資訊。

使用 的單一區域容器機群 FlexMatch

此案例會將您的遊戲伺服器部署到容器機群、設定遊戲工作階段置放，以及設定FlexMatch配對。當您準備好開始為託管解決方案設計自訂配對建構器時，此案例非常有用。使用此案例來建立此解決方案的基本資源，您可以視需要稍後進行自訂。它會部署下列資源：

- Amazon GameLift Servers 容器群組定義，說明如何在容器機群上部署和執行容器映像。
- Amazon GameLift Servers 安裝並執行遊戲伺服器容器的容器機群（隨需），以及別名。
- FlexMatch 配對組態和配對規則集，以接受玩家請求並形成配對。
- Amazon GameLift Servers 透過尋找最佳的託管資源（根據可行性、成本、玩家延遲等）並啟動遊戲工作階段，滿足對提議配對的請求的遊戲工作階段佇列。
- Amazon Cognito 使用者集區和用戶端可讓玩家驗證和啟動遊戲。
- 將使用者集區與 API 連結APIs Gateway 授權方。
- Web 存取控制清單 (ACL)，用於調節對 API Gateway 的過多玩家呼叫。
- 代表遊戲用戶端向Amazon GameLift Servers服務提出請求的後端服務，例如請求遊戲工作階段和加入遊戲：
 - API Gateway + Lambda 函數，讓玩家請求遊戲工作階段槽。StartMatchmaking() 如果沒有可用的開放插槽，此函數會呼叫。
 - API Gateway + Lambda 函數，讓玩家取得其遊戲請求的連線資訊。
- DynamoDB 資料表可存放玩家的配對票證和遊戲工作階段資訊。
- Amazon SNS 主題 + Lambda 函數來處理 GameSessionQueue 事件。

部署容器機群

當您的機群組態完成時，請選擇部署容器機群按鈕以開始部署。當外掛程式建立容器映像並將其推送至 ECR、為容器機群佈建託管資源、為選取的託管解決方案案例部署機群和其他 AWS 資源時，此程序可能需要幾分鐘的時間。

開始部署時，您可以追蹤每個步驟的進度。根據您的組態，這些步驟可能包含下列項目：

- 設定容器映像
- 建立 Amazon ECR 儲存庫
- 建置映像並推送至 Amazon ECR
- 建立容器群組定義
- 建立容器機群

如需更詳細的部署資訊，請選擇 管理主控台 中的 AWS 檢視。當容器機群達到作用中狀態時，機群會主動執行具有準備好託管遊戲工作階段之伺服器程序的容器。

當部署完成時，您會有一個工作容器機群，已準備好託管遊戲工作階段並接受玩家連線。

您無法停止進行中的部署。如果部署進入錯誤狀態或失敗，您可以使用重設部署選項重新開始。

啟動用戶端

此時，您已完成所有任務，以啟動和播放使用託管的多玩家遊戲 Amazon GameLift Servers。若要玩遊戲，請選擇啟動用戶端以啟動遊戲用戶端的本機執行個體。

- 如果您部署了單一機群案例，請使用一個玩家開啟遊戲用戶端的一個執行個體，然後輸入伺服器地圖以四處移動。您可以開啟遊戲用戶端的第二個執行個體，將第二個玩家新增至相同的伺服器遊戲地圖。
- 如果您部署了 FlexMatch 案例，託管解決方案會等待至少兩個遊戲用戶端提出配對請求。使用一個玩家開啟遊戲用戶端的至少兩個執行個體。兩個玩家將會進行配對，並提示他們加入配對的遊戲工作階段。

更新容器機群

如果您已成功部署受管容器託管解決方案，您可以使用更新部署功能。此選項可讓您更新已部署容器機群的組態設定，而不必建立新的機群。

更新部署時，您可以使用不同的遊戲伺服器建置來部署容器映像、變更 Amazon ECR 儲存庫、選擇不同的部署案例，以及自訂選用的組態設定。

當您準備好部署變更時，請選擇更新。部署更新所需的時間類似於完整部署。如需詳細部署資訊，請選擇 AWS 管理主控台中的檢視。

清除已部署的資源

最佳實務是在不再需要受管容器解決方案時，立即清理這些解決方案 AWS 的資源。如果您不移除這些資源，則可能會繼續產生這些資源的成本。

刪除下列資源：

- 受管容器資源堆疊。此堆疊中的資源取決於您選擇的部署案例。若要刪除整個堆疊，請使用 AWS CloudFormation 主控台。從 Amazon GameLift Servers 外掛程式產生的堆疊使用以下命名慣例：GameLiftPluginForUnity-{GameName}-Containers。在外掛程式中啟動新的受管容器部署之前，請等待堆疊刪除程序完成。如需詳細資訊，請參閱[從 CloudFormation 主控台刪除堆疊](#)。
- Amazon ECR 儲存庫。如果您使用外掛程式為容器映像建立儲存庫，您可能想要刪除任何不再需要的儲存庫。在重設受管容器部署之前，您不需要刪除儲存庫。如果您更新或重設部署，外掛程式將自動使用相同的儲存庫，除非被指示使用另一個儲存庫。如需詳細資訊，請參閱[刪除 Amazon ECR 中的私有儲存庫](#)。

Amazon GameLift ServersUnity 的外掛程式（伺服器 SDK 4.x）

Note

本主題提供 Unity 舊版 Amazon GameLift Servers 外掛程式的資訊。1.0.0 版 (2021 年發行) 使用適用於 4.x Amazon GameLift Servers 或更早版本的伺服器 SDK。如需使用伺服器 SDK 5.x 且支援 Amazon GameLift ServersAnywhere 的最新版本外掛程式的文件，請參閱 [Amazon GameLift ServersUnity 的外掛程式（伺服器 SDK 5.x）](#)。

Amazon GameLift Servers 提供工具，讓您的多玩家遊戲伺服器準備好在 上執行 Amazon GameLift Servers。Unity 的 Amazon GameLift Servers 外掛程式可讓您更輕鬆地整合 Amazon GameLift Servers 到 Unity 遊戲專案，並部署雲端託管 Amazon GameLift Servers 的資源。使用 Unity 的外掛程式來存取 Amazon GameLift Servers APIs 和部署 AWS CloudFormation 常見遊戲案例的範本。

設定外掛程式之後，您可以在 GitHub 上嘗試 [Amazon GameLift Servers Unity 範例](#)。

主題

- [Amazon GameLift Servers 與 Unity 遊戲伺服器專案整合](#)
- [Amazon GameLift Servers 與 Unity 遊戲用戶端專案整合](#)
- [安裝和設定 外掛程式](#)
- [在本機測試您的遊戲](#)
- [部署案例](#)
- [將遊戲與 Unity Amazon GameLift Servers 中的 整合](#)
- [匯入並執行範例遊戲](#)

Amazon GameLift Servers 與 Unity 遊戲伺服器專案整合

Note

本主題提供 Unity 舊版 Amazon GameLift Servers 外掛程式的資訊。1.0.0 版 (2021 年發行) 使用適用於 4.x Amazon GameLift Servers 或更早版本的伺服器 SDK。如需使用伺服器 SDK 5.x 且支援 Amazon GameLift Servers Anywhere 的最新版本外掛程式的文件，請參閱 [Amazon GameLift Servers Unity 的外掛程式 \(伺服器 SDK 5.x\)](#)。

本主題可協助您準備自訂遊戲伺服器，以便在 上託管 Amazon GameLift Servers。遊戲伺服器必須能夠 Amazon GameLift Servers 通知其狀態、在出現提示時啟動和停止遊戲工作階段，以及執行其他任務。如需詳細資訊，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

先決條件

在整合遊戲伺服器之前，請先完成下列任務：

- [設定 的 IAM 服務角色 Amazon GameLift Servers](#)
- [安裝和設定 外掛程式](#)

設定新的伺服器程序

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

設定與 的通訊，Amazon GameLift Servers 並回報伺服器程序已準備好託管遊戲工作階段。

1. 呼叫 來初始化伺服器 SDKInitSDK()。
2. 若要讓伺服器準備好接受遊戲工作階段，ProcessReady() 請使用連線連接埠和遊戲工作階段位置詳細資訊呼叫。包含 Amazon GameLift Servers 服務叫用的回呼函數名稱，例如 OnGameSession()、OnGameSessionUpdate()、OnProcessTerminate()、OnHealthCheck()。Amazon GameLift Servers 可能需要幾分鐘才能提供回呼。
3. Amazon GameLift Servers 會將伺服器程序的狀態更新為 ACTIVE。
4. Amazon GameLift Servers onHealthCheck 會定期呼叫。

下列程式碼範例示範如何使用 設定簡單的伺服器程序 Amazon GameLift Servers。

```
//initSDK
var initSDKOutcome = GameLiftServerAPI.InitSDK();

//processReady
// Set parameters and call ProcessReady
var processParams = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    // Examples of log and error files written by the game server
    new LogParameters(new List<string>()
        {
            "C:\\game\\logs",
            "C:\\game\\error"
        })
);

var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);

// Implement callback functions
void OnGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```



```
void OnProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    var ProcessEndingOutcome = GameLiftServerAPI.ProcessEnding();
}

bool OnHealthCheck()
{
    bool isHealthy;
    // complete health evaluation within 60 seconds and set health
    return isHealthy;
}
```

啟動遊戲工作階段

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

遊戲初始化完成後，您可以啟動遊戲工作階段。

1. 實作回呼函數 `onStartGameSession`。會 Amazon GameLift Servers 叫用此方法，在伺服器程序上啟動新的遊戲工作階段，並接收玩家連線。
2. 若要啟用遊戲工作階段，請呼叫 `ActivateGameSession()`。如需 SDK 的詳細資訊，請參閱 [適用於 Amazon GameLift Servers 4.x 的 C# 伺服器 SDK -- 動作](#)。

下列程式碼範例說明如何使用 啟動遊戲工作階段 Amazon GameLift Servers。

```
void OnStartGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    ...
    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

結束遊戲工作階段

Note

本主題參考 Unity 1.0.0 版的Amazon GameLift Servers外掛程式，其使用伺服器 SDK 4.x 或更早版本。

Amazon GameLift Servers 遊戲工作階段結束時通知。最佳實務是在遊戲工作階段完成後關閉伺服器程序，以回收和重新整理託管資源。

1. 設定名為的函數onProcessTerminate來接收來自的請求Amazon GameLift Servers並呼叫ProcessEnding()。
2. 程序狀態會變更為 TERMINATED。

下列範例說明如何結束遊戲工作階段的程序。

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();

if (processReadyOutcome.Success)
    Environment.Exit(0);

// otherwise, exit with error code
Environment.Exit(errorCode);
```

建立伺服器建置並上傳至 Amazon GameLift Servers

Note

本主題參考 Unity 1.0.0 版的Amazon GameLift Servers外掛程式，其使用伺服器 SDK 4.x 或更早版本。

將遊戲伺服器與整合之後Amazon GameLift Servers，請將建置檔案上傳至機群，讓 Amazon GameLift Servers可以將其部署以進行遊戲託管。如需如何將伺服器上傳至的詳細資訊Amazon GameLift Servers，請參閱 [部署用於Amazon GameLift Servers託管的自訂伺服器建置](#)。

Amazon GameLift Servers 與 Unity 遊戲用戶端專案整合

Note

本主題提供 Unity 舊版Amazon GameLift Servers外掛程式的資訊。1.0.0 版 (2021 年發行) 使用適用於 4.x Amazon GameLift Servers 或更早版本的伺服器 SDK。如需使用伺服器 SDK 5.x 且支援 Amazon GameLift ServersAnywhere 的最新版本外掛程式的文件，請參閱 [Amazon GameLift ServersUnity 的外掛程式 \(伺服器 SDK 5.x\)](#)。

本主題可協助您設定遊戲用戶端，透過後端服務連線至Amazon GameLift Servers託管遊戲工作階段。使用 Amazon GameLift Servers APIs啟動配對、請求遊戲工作階段放置等。

將程式碼新增至後端服務專案，以允許與服務通訊Amazon GameLift Servers。後端服務會處理與 GameLift 服務的所有遊戲用戶端通訊。如需後端服務的詳細資訊，請參閱 。

後端伺服器會處理下列遊戲用戶端任務：

- 為您的玩家自訂身分驗證。
- 向 Amazon GameLift Servers 服務請求有關作用中遊戲工作階段的資訊。
- 建立新的遊戲工作階段。
- 將玩家新增至現有的遊戲工作階段。
- 從現有的遊戲工作階段中移除玩家。

主題

- [先決條件](#)
- [初始化遊戲用戶端](#)
- [在特定機群上建立遊戲工作階段](#)
- [將玩家新增至遊戲工作階段](#)
- [從遊戲工作階段中移除玩家](#)

先決條件

在設定與Amazon GameLift Servers用戶端的遊戲伺服器通訊之前，請先完成下列任務：

- [設定 AWS 帳戶](#)

- [安裝和設定 外掛程式](#)
- [Amazon GameLift Servers 與 Unity 遊戲伺服器專案整合](#)
- [使用 設定託管機群 Amazon GameLift Servers](#)

初始化遊戲用戶端

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

新增程式碼以初始化遊戲用戶端。啟動時執行此程式碼，這是其他 Amazon GameLift Servers 函數的必要項目。

1. 初始化 AmazonGameLiftClient。AmazonGameLiftClient 使用預設用戶端組態或自訂組態呼叫。如需如何設定用戶端的詳細資訊，請參閱 [在後端服務 Amazon GameLift Servers 上設定](#)。
2. 為每個玩家產生唯一的玩家 ID，以連接到遊戲工作階段。如需更多資訊，請參閱 [產生玩家 IDs](#)。

下列範例示範如何設定 Amazon GameLift Servers 用戶端。

```
public class GameLiftClient
{
    private GameLift gl;
    //A sample way to generate random player IDs.
    bool includeBrackets = false;
    bool includeDashes = true;
    string playerId = AZ::Uuid::CreateRandom().ToString<string>(includeBrackets,
includeDashes);

    private Amazon.GameLift.Model.PlayerSession psession = null;
    public AmazonGameLiftClient aglc = null;

    public void CreateGameLiftClient()
    {
        //Access Amazon GameLift Servers service by setting up a configuration.
        //The default configuration specifies a location.
        var config = new AmazonGameLiftConfig();
        config.RegionEndpoint = Amazon.RegionEndpoint.USEast1;
```

```
CredentialProfile profile = null;
var nscf = new SharedCredentialsFile();
nscf.TryGetProfile(profileName, out profile);
AWSCredentials credentials = profile.GetAWSCredentials(null);
//Initialize Amazon GameLift Servers Client with default client
configuration.
aglc = new AmazonGameLiftClient(credentials, config);

}
}
```

在特定機群上建立遊戲工作階段

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

加入用於在已部署的機群中啟動新遊戲工作階段並使其可供玩家加入的程式碼。在 Amazon GameLift Servers 建立新的遊戲工作階段並傳回 之後 GameSession，您可以將玩家新增至該工作階段。

- 提出新遊戲工作階段的請求。
 - 如果您的遊戲使用機群，CreateGameSession() 請使用機群或別名 ID、工作階段名稱，以及遊戲的並行玩家數量上限來呼叫。
 - 如果您的遊戲使用佇列，請呼叫 StartGameSessionPlacement()。

下列範例示範如何建立遊戲工作階段。

```
public Amazon.GameLift.Model.GameSession()
{
    var cgsreq = new Amazon.GameLift.Model.CreateGameSessionRequest();
    //A unique identifier for the alias with the fleet to create a game session in.
    cgsreq.AliasId = aliasId;
    //A unique identifier for a player or entity creating the game session
    cgsreq.CreatorId = playerId;
    //The maximum number of players that can be connected simultaneously to the game
    session.
}
```

```
cgsreq.MaximumPlayerSessionCount = 4;

//Prompt an available server process to start a game session and retrieves
connection information for the new game session
Amazon.GameLift.Model.CreateGameSessionResponse cgsres =
aglc.CreateGameSession(cgsreq);
string gsid = cgsres.GameSession != null ? cgsres.GameSession.GameSessionId : "N/A";
Debug.Log((int)cgsres.HttpStatusCode + " GAME SESSION CREATED: " + gsid);
return cgsres.GameSession;
}
```

將玩家新增至遊戲工作階段

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

在 Amazon GameLift Servers 建立新的遊戲工作階段並傳回 `GameSession` 物件之後，您可以將玩家新增至該工作階段。

1. 透過建立新的玩家工作階段，在遊戲工作階段中預留玩家位置。使用 `CreatePlayerSession` 或 `CreatePlayerSessions` 搭配遊戲工作階段 ID 和每個玩家的唯一 ID。
2. 連線至遊戲工作階段。擷取 `PlayerSession` 物件以取得遊戲工作階段的連線資訊。您可以使用此資訊建立與伺服器程序的直接連線：
 - a. 使用指定的連接埠，以及伺服器程序的 DNS 名稱或 IP 地址。
 - b. 使用機群的 DNS 名稱和連接埠。如果您的機群已啟用 TLS 憑證產生，則需要 DNS 名稱和連接埠。
 - c. 參考玩家工作階段 ID。如果您的遊戲伺服器驗證傳入的玩家連線，則需要玩家工作階段 ID。

下列範例示範如何在遊戲工作階段中預留玩家位置。

```
public Amazon.GameLift.Model.PlayerSession
CreatePlayerSession(Amazon.GameLift.Model.GameSession gsession)
{
    var cpsreq = new Amazon.GameLift.Model.CreatePlayerSessionRequest();
```

```
cpsreq.GameSessionId = gsession.GameSessionId;
//Specify game session ID.
cpsreq.PlayerId = playerId;
//Specify player ID.
Amazon.GameLift.Model.CreatePlayerSessionResponse cpsres =
aglc.CreatePlayerSession(cpsreq);
string psid = cpsres.PlayerSession != null ? cpsres.PlayerSession.PlayerSessionId :
"N/A";
return cpsres.PlayerSession;
}
```

下列程式碼說明如何將玩家與遊戲工作階段連線。

```
public bool ConnectPlayer(int playerId, string playerSessionId)
{
    //Call ConnectPlayer with player ID and player session ID.
    return server.ConnectPlayer(playerId, playerSessionId);
}
```

從遊戲工作階段中移除玩家

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

您可以在玩家離開遊戲時，將玩家從遊戲工作階段中移除。

1. 通知 Amazon GameLift Servers 服務玩家已中斷與伺服器程序的連線。RemovePlayerSession 使用玩家的工作階段 ID 呼叫。
2. 確認 RemovePlayerSession 傳回 Success。然後，會將玩家位置 Amazon GameLift Servers 變更為可用，Amazon GameLift Servers 可指派給新的玩家。

下列範例說明如何移除玩家工作階段。

```
public void DisconnectPlayer(int playerId)
{
    //Receive the player session ID.
    string playerSessionId = playerSessions[playerId];
}
```



```
var outcome = GameLiftServerAPI.RemovePlayerSession(playerSessionId);
if (outcome.Success)
{
    Debug.Log (":) PLAYER SESSION REMOVED");
}
else
{
    Debug.Log(":(PLAYER SESSION REMOVE FAILED. RemovePlayerSession()
    returned " + outcome.Error.ToString());
}
}
```

安裝和設定 外掛程式

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

本節說明如何下載、安裝和設定 Unity 的 Amazon GameLift Servers 外掛程式 1.0.0 版。

先決條件

- Unity for Windows 2019.4 LTS、Windows 2020.3 LTS 或 Unity for MacOS
- Java 的目前版本
- .NET 4.x 的目前版本

下載並安裝 Unity 的 外掛程式

1. 下載 Unity 的 Amazon GameLift Servers 外掛程式。您可以在 [Amazon GameLift Servers Unity 儲存庫的外掛程式](#) 頁面上找到最新版本。在 [最新版本](#) 下，選擇資產，然後下載 `com.amazonaws.gamelift-version.tgz` 檔案。
2. 啟動 Unity 並選擇專案。
3. 在頂端導覽列的視窗下，選擇套件管理員：
4. 在套件管理員索引標籤下，選擇 +，然後選擇從 tarball 新增套件...

5. 在磁碟上選取套件視窗中，導覽至 `com.amazonaws.gamelift` 資料夾，選擇檔案 `com.amazonaws.gamelift-version.tgz`，然後選擇開啟：
6. Unity 載入外掛程式後，Amazon GameLift Servers 會在 Unity 功能表中顯示為新項目。安裝和重新編譯指令碼可能需要幾分鐘的時間。Amazon GameLift Servers 外掛程式設定索引標籤會自動開啟。
7. 在 SDK 窗格中，選擇使用 .NET 4.x。

設定後，狀態會從未設定變更為已設定。

在本機測試您的遊戲

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

使用 Amazon GameLift Servers Local 在 Amazon GameLift Servers 本機裝置上執行。您可以使用 Amazon GameLift Servers Local 在幾秒鐘內驗證程式碼變更，無需網路連線。

設定本機測試

1. 在 Unity 的外掛程式視窗中，選擇測試索引標籤。
2. 在測試窗格中，選擇下載 Amazon GameLift Servers 本機。Unity 的外掛程式會開啟瀏覽器視窗，並將 `GameLift_06_03_2021.zip` 檔案下載到您的下載資料夾。

下載包含 C# 伺服器 SDK、.NET 來源檔案，以及與 Unity 相容的 .NET 元件。

3. 解壓縮所下載的 `GameLift_06_03_2021.zip` 檔案。
4. 在 Amazon GameLift Servers 外掛程式設定視窗中，選擇 Amazon GameLift Servers 本機路徑，導覽至解壓縮資料夾，選擇檔案 `GameLiftLocal.jar`，然後選擇開啟。

設定時，本機測試狀態會從未設定變更為已設定。

5. 驗證 JRE 的狀態。如果狀態為未設定，請選擇下載 JRE 並安裝建議的 Java 版本。

安裝和設定 Java 環境之後，狀態會變更為已設定。

執行您的本機遊戲

1. 在 Unity 的外掛程式索引標籤中，選擇測試索引標籤。
2. 在測試窗格中，選擇開啟本機測試 UI。
3. 在本機測試視窗中，指定伺服器可執行檔路徑。選取 ... 以選取伺服器應用程式的路徑和可執行檔名稱。
4. 在本機測試視窗中，指定 GL 本機連接埠。
5. 選擇部署和執行以部署和執行伺服器。
6. 若要停止遊戲伺服器，請選擇停止或關閉遊戲伺服器視窗。

部署案例

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

案例使用 AWS CloudFormation 範本來建立部署遊戲雲端託管解決方案所需的資源。本節說明提供的案例及其使用 Amazon GameLift Servers 方式。

先決條件

若要部署案例，您需要 Amazon GameLift Servers 服務的 IAM 角色。如需如何為 建立角色的資訊 Amazon GameLift Servers，請參閱 [設定 AWS 帳戶](#)。

每個案例都需要下列資源的許可：

- Amazon GameLift Servers
- Amazon S3
- AWS CloudFormation
- API Gateway
- AWS Lambda
- AWS WAFV2
- Amazon Cognito

案例

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

Unity 的 Amazon GameLift Servers 外掛程式包含下列案例：

僅限驗證

此案例會建立遊戲後端服務，在沒有遊戲伺服器功能的情況下執行玩家身分驗證。範本會在您的帳戶中建立下列資源：

- 存放玩家身分驗證資訊的 Amazon Cognito 使用者集區。
- Amazon API Gateway REST 端點後端 AWS Lambda 處理常式，可啟動遊戲並檢視遊戲連線資訊。

單一區域機群

此案例會使用單一 Amazon GameLift Servers 機群建立遊戲後端服務。它會建立下列資源：

- 供玩家驗證和啟動遊戲的 Amazon Cognito 使用者集區。
- 用來搜尋機群上具有開啟玩家位置之現有遊戲工作階段的 AWS Lambda 處理常式。如果找不到開啟的插槽，則會建立新的遊戲工作階段。

具有佇列和自訂配對建構器的多區域機群

此案例使用 Amazon GameLift Servers 佇列和自訂配對建構器來組成配對，將等待集區中最舊的玩家分組在一起。它會建立下列資源：

- 發佈訊息的 Amazon Simple Notification Service Amazon GameLift Servers 主題。如需 SNS 主題和通知的詳細資訊，請參閱 [設定遊戲工作階段放置的事件通知](#)。
- 由訊息呼叫的 Lambda 函數，用於傳達置放和遊戲連線詳細資訊。
- 用於存放置放和遊戲連線詳細資訊的 Amazon DynamoDB 資料表。會 `GetGameConnection` 呼叫從此資料表讀取並將連線資訊傳回至遊戲用戶端。

具有佇列和自訂配對建構器的 Spot 機群

此案例使用Amazon GameLift Servers佇列和自訂配對建構器來形成配對，並設定三個機群。它會建立下列資源：

- 包含不同執行個體類型的兩個 Spot 機群，可為 Spot 無法使用提供耐用性。
- 做為其他 Spot 機群備份的隨需機群。如需設計機群的詳細資訊，請參閱 [自訂您的 Amazon GameLift Servers EC2 受管機群](#)。
- 保持伺服器可用性高且成本低的Amazon GameLift Servers佇列。如需佇列的詳細資訊和最佳實務，請參閱 [自訂遊戲工作階段佇列](#)。

FlexMatch

此案例使用受管配對服務 FlexMatch，將遊戲玩家配對在一起。如需的詳細資訊FlexMatch，請參閱[什麼是 Amazon GameLift Servers FlexMatch](#)。此案例會建立下列資源：

- 在收到StartGame請求後建立配對票證的 Lambda 函數。
- 用來接聽FlexMatch相符事件的個別 Lambda 函數。

為了避免您的產生不必要的費用 AWS 帳戶，請在完成使用後移除每個案例建立的資源。刪除對應的AWS CloudFormation 堆疊。

更新 AWS 登入資料

Note

本主題參考 Unity 1.0.0 版的Amazon GameLift Servers外掛程式，其使用伺服器 SDK 4.x 或更早版本。

Unity 的 Amazon GameLift Servers外掛程式需要安全登入資料才能部署案例。您可以建立新的登入資料或使用現有的登入資料。

如需設定登入資料的詳細資訊，請參閱[了解並取得您的 AWS 登入資料](#)。

更新 AWS 登入資料

1. 在 Unity 的 Unity 外掛程式索引標籤中，選擇部署索引標籤。
2. 在部署窗格中，選擇AWS 登入資料。
3. 您可以建立新的 AWS 登入資料或選擇現有的登入資料。

- 若要建立登入資料，請選擇建立新的登入資料設定檔，然後指定新設定檔名稱、AWS 存取金鑰 ID、AWS 私密金鑰和 AWS 區域。
 - 若要選擇現有的登入資料，請選擇選擇現有的登入資料設定檔，然後選擇設定檔名稱 和 AWS 區域。
4. 在更新 AWS 登入資料視窗中，選擇更新登入資料設定檔。

更新帳戶引導

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，該外掛程式使用伺服器 SDK 4.x 或更早版本。

引導位置是在部署期間使用的 Amazon S3 儲存貯體。它用於存放遊戲伺服器資產和其他相依性。AWS 區域 您為儲存貯體選擇的 必須與您將用於案例部署的區域相同。

如需 Amazon S3 儲存貯體的詳細資訊，請參閱[建立、設定和使用 Amazon Simple Storage Service 儲存貯體](#)。

更新帳戶引導位置

1. 在 Unity 的 Unity 外掛程式索引標籤中，選擇部署索引標籤。
2. 在部署窗格中，選擇更新帳戶引導。
3. 在帳戶引導視窗中，您可以選擇現有的 Amazon S3 儲存貯體或建立新的 Amazon S3 儲存貯體：
 - 若要選擇現有的儲存貯體，請選擇選擇現有的 Amazon S3 儲存貯體並更新以儲存您的選擇。
 - 選擇建立新的 Amazon S3 儲存貯體以建立新的 Amazon Simple Storage Service 儲存貯體，然後選擇政策。此政策會指定 Amazon S3 儲存貯體的過期時間。選擇建立以建立儲存貯體。

部署遊戲案例

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，該外掛程式使用伺服器 SDK 4.x 或更早版本。

您可以使用 案例來測試您的遊戲Amazon GameLift Servers。每個案例都使用 AWS CloudFormation 範本來建立具有所需資源的堆疊。大多數案例都需要遊戲伺服器可執行檔和建置路徑。當您部署案例時，會將遊戲資產Amazon GameLift Servers複製到引導位置，做為部署的一部分。

您必須設定 AWS 登入資料和 AWS 帳戶引導才能部署案例。

部署案例

1. 在 Unity 的 Unity 外掛程式索引標籤中，選擇部署索引標籤。
2. 在部署窗格中，選擇開啟部署 UI。
3. 在部署視窗中，選擇案例。
4. 輸入遊戲名稱。它必須獨一無二。當您部署案例時，遊戲名稱是 AWS CloudFormation 堆疊名稱的一部分。
5. 選擇遊戲伺服器建置資料夾路徑。建置資料夾路徑會指向包含伺服器可執行檔和相依性的資料夾。
6. 選擇遊戲伺服器建置 .exe 檔案路徑。建置可執行檔路徑指向遊戲伺服器可執行檔。
7. 選擇開始部署以開始部署案例。您可以在目前狀態下的部署視窗中遵循更新的狀態。部署可能需要幾分鐘的時間。
8. 當案例完成部署時，目前狀態會更新以包含 Cognito 用戶端 ID 和 API Gateway 端點，您可以將其複製並貼到遊戲中。
9. 若要更新遊戲設定，請在 Unity 功能表中，選擇移至用戶端連線設定。這會在 Unity 畫面右側顯示 Inspector 索引標籤。
10. 取消選取本機測試模式。
11. 輸入 API Gateway 端點和 Cognito 用戶端 ID。選擇 AWS 區域 您用於案例部署的相同。然後，您可以使用部署的案例資源重建和執行遊戲用戶端。

刪除案例建立的資源

Note

本主題參考 Unity 1.0.0 版的Amazon GameLift Servers外掛程式，該外掛程式使用伺服器 SDK 4.x 或更早版本。

若要刪除為案例建立的資源，請刪除對應的 AWS CloudFormation 堆疊。

刪除案例建立的資源

1. 在 Unity Deployment 的 Amazon GameLift Servers 外掛程式視窗中，選取檢視 AWS CloudFormation 主控台以開啟 AWS CloudFormation 主控台。
2. 在 AWS CloudFormation 主控台中，選擇 Stacks，然後選擇包含部署期間指定遊戲名稱的堆疊。
3. 選擇刪除以刪除堆疊。刪除堆疊可能需要幾分鐘的時間。在 AWS CloudFormation 刪除案例使用的堆疊後，其狀態會變更為 ROLLBACK_COMPLETE。

將遊戲與 Unity Amazon GameLift Servers 中的 整合

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，該外掛程式使用伺服器 SDK 4.x 或更早版本。

完成下列任務 Amazon GameLift Servers，將您的 Unity 遊戲與 整合：

- [Amazon GameLift Servers 與 Unity 遊戲伺服器專案整合](#)
- [Amazon GameLift Servers 與 Unity 遊戲用戶端專案整合](#)

下圖顯示整合遊戲的範例流程。在圖表中，具有遊戲伺服器的機群會部署到 Amazon GameLift Servers。遊戲用戶端會與遊戲伺服器通訊，其會與 通訊 Amazon GameLift Servers。

匯入並執行範例遊戲

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，該外掛程式使用伺服器 SDK 4.x 或更早版本。

Unity 的 Amazon GameLift Servers 外掛程式包含範例遊戲，可用來探索整合遊戲與 的基本概念 Amazon GameLift Servers。在本節中，您會建置遊戲用戶端和遊戲伺服器，然後使用 Amazon GameLift Servers Local 在本機進行測試。

先決條件

- [設定 AWS 帳戶](#)
- [安裝和設定 外掛程式](#)

建置並執行範例遊戲伺服器

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，該外掛程式使用伺服器 SDK 4.x 或更早版本。

設定範例遊戲的遊戲伺服器檔案。

1. 在 Unity 的選單上，選擇 Amazon GameLift Servers，然後選擇匯入範例遊戲。
2. 在匯入範例遊戲視窗中，選擇匯入以匯入遊戲、其資產和相依性。
3. 建置遊戲伺服器。在 Unity 的功能表中，選擇 Amazon GameLift Servers，然後選擇套用 Windows 範例伺服器建置設定或套用 MacOS 範例伺服器建置設定。設定遊戲伺服器設定後，Unity 會重新編譯資產。
4. 在 Unity 的功能表中，選擇檔案，然後選擇建置。選擇伺服器建置，選擇建置，然後選擇專門用於伺服器檔案的建置資料夾。

Unity 會建置範例遊戲伺服器，將可執行檔和必要的資產放在指定的建置資料夾中。

建置並執行範例遊戲用戶端

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，該外掛程式使用伺服器 SDK 4.x 或更早版本。

設定範例遊戲的遊戲用戶端檔案。

1. 在 Unity 的選單上，選擇 Amazon GameLift Servers，然後選擇套用 Windows 範例用戶端建置設定或套用 MacOS 範例用戶端建置設定。設定遊戲用戶端設定後，Unity 會重新編譯資產。

2. 在 Unity 的功能表中，選取移至用戶端設定。這會在 Unity 畫面右側顯示 Inspector 標籤。在 Amazon GameLift Servers 用戶端設定索引標籤中，選取本機測試模式。
3. 建置遊戲用戶端。在 Unity 的功能表中，選擇檔案。確認未核取伺服器建置，選擇建置，然後特別為用戶端檔案選取建置資料夾。

Unity 會建置範例遊戲用戶端，將可執行檔和必要的資產放在指定的用戶端建置資料夾中。

4. 您尚未建置遊戲伺服器 and 用戶端。在後續步驟中，您會執行遊戲，並查看其與的互動方式 Amazon GameLift Servers。

在本機測試範例遊戲

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，該外掛程式使用伺服器 SDK 4.x 或更早版本。

執行您使用 Amazon GameLift Servers Local 匯入的範例遊戲。

1. 啟動遊戲伺服器。在 Unity 的 Unity 外掛程式索引標籤中，選擇部署索引標籤。
2. 在測試窗格中，選擇開啟本機測試 UI。
3. 在本機測試視窗中，指定遊戲伺服器 .exe 檔案路徑。路徑必須包含可執行檔名稱。例如 C:/MyGame/GameServer/MyGameServer.exe。
4. 選擇部署和執行。Unity 的外掛程式會啟動遊戲伺服器，並開啟 Amazon GameLift Servers 本機日誌視窗。視窗包含日誌訊息，包括在遊戲伺服器與 Amazon GameLift Servers Local 之間傳送的訊息。
5. 啟動遊戲用戶端。使用範例遊戲用戶端尋找建置位置，然後選擇可執行檔。
6. 在 Amazon GameLift Servers 範例遊戲中，提供電子郵件和密碼，然後選擇登入。電子郵件和密碼未經驗證或使用。
7. 在 Amazon GameLift Servers 範例遊戲中，選擇開始。遊戲用戶端會尋找遊戲工作階段。如果找不到工作階段，則會建立一個工作階段。遊戲用戶端接著會啟動遊戲工作階段。您可以在日誌中查看遊戲活動。

遊戲伺服器日誌範例

...

```

2021-09-15T19:55:3495 PID:20728 Log :) GAMELIFT AWAKE
2021-09-15T19:55:3512 PID:20728 Log :) I AM SERVER
2021-09-15T19:55:3514 PID:20728 Log :) GAMELIFT StartServer at port 33430.
2021-09-15T19:55:3514 PID:20728 Log :) SDK VERSION: 4.0.2
2021-09-15T19:55:3556 PID:20728 Log :) SERVER IS IN A GAMELIFT FLEET
2021-09-15T19:55:3577 PID:20728 Log :) PROCESSREADY SUCCESS.
2021-09-15T19:55:3577 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
...
2021-09-15T19:55:3634 PID:20728 Log :) GAMELOGIC AWAKE
2021-09-15T19:55:3635 PID:20728 Log :) GAMELOGIC START
2021-09-15T19:55:3636 PID:20728 Log :) LISTENING ON PORT 33430
2021-09-15T19:55:3636 PID:20728 Log SERVER: Frame: 0 HELLO WORLD!
...
2021-09-15T19:56:2464 PID:20728 Log :) GAMELIFT SESSION REQUESTED
2021-09-15T19:56:2468 PID:20728 Log :) GAME SESSION ACTIVATED
2021-09-15T19:56:3578 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
2021-09-15T19:57:3584 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
2021-09-15T19:58:0334 PID:20728 Log SERVER: Frame: 8695 Connection accepted: playerId
0 joined
2021-09-15T19:58:0335 PID:20728 Log SERVER: Frame: 8696 Connection accepted: playerId
1 joined
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerId 0 Msg:
CONNECT: server IP localhost
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 0:CONNECT:
server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 0
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerId 1 Msg:
CONNECT: server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 1:CONNECT:
server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 1

```

範例Amazon GameLift Servers本機日誌

```

12:55:26,000 INFO || - [SocketIOServer] main - Session store / pubsub factory used:
MemoryStoreFactory (local session store only)
12:55:28,092 WARN || - [ServerBootstrap] main - Unknown channel option 'SO_LINGER' for
channel '[id: 0xe23d0a14]'
12:55:28,101 INFO || - [SocketIOServer] nioEventLoopGroup-2-1 - SocketIO server
started at port: 5757
12:55:28,101 INFO || - [SDKConnection] main - GameLift SDK server (communicates with
your game server) has started on http://localhost:5757

```

```

12:55:28,120 INFO || - [SdkWebSocketServer] WebSocketSelector-20 - WebSocket Server
started on address localhost/127.0.0.1:5759
12:55:28,166 INFO || - [StandAloneServer] main - GameLift Client server (listens for
GameLift client APIs) has started on http://localhost:8080
12:55:28,179 INFO || - [StandAloneServer] main - GameLift server sdk http listener has
started on http://localhost:5758
12:55:35,453 INFO || - [SdkWebSocketServer] WebSocketWorker-12 - onOpen
socket: /?pID=20728&sdkVersion=4.0.2&sdkLanguage=CSharp and handshake /?
pID=20728&sdkVersion=4.0.2&sdkLanguage=CSharp
12:55:35,551 INFO || - [HostProcessManager] WebSocketWorker-12 - client connected with
pID 20728
12:55:35,718 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
GameLift API to use: ProcessReady for pId 20728
12:55:35,718 INFO || - [ProcessReadyHandler] GameLiftSdkHttpHandler-thread-0 -
Received API call for processReady from 20728
12:55:35,738 INFO || - [ProcessReadyHandler] GameLiftSdkHttpHandler-thread-0 -
onProcessReady data: port: 33430
12:55:35,739 INFO || - [HostProcessManager] GameLiftSdkHttpHandler-thread-0 -
Registered new process with pId 20728
12:55:35,789 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
GameLift API to use: ReportHealth for pId 20728
12:55:35,790 INFO || - [ReportHealthHandler] GameLiftSdkHttpHandler-thread-0 -
Received API call for ReportHealth from 20728
12:55:35,794 INFO || - [ReportHealthHandler] GameLiftSdkHttpHandler-thread-0 -
ReportHealth data: healthStatus: true
12:56:24,098 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
GameLift.DescribeGameSessions
12:56:24,119 INFO || - [DescribeGameSessionsDispatcher] Thread-12 - Received API call
to describe game sessions with input: {"FleetId":"fleet-123"}
12:56:24,241 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
GameLift.CreateGameSession
12:56:24,242 INFO || - [CreateGameSessionDispatcher] Thread-12 - Received API call to
create game session with input: {"FleetId":"fleet-123","MaximumPlayerSessionCount":4}
12:56:24,265 INFO || - [HostProcessManager] Thread-12 - Reserved process:
20728 for gameSession: arn:aws:gamelift:local::gamesession/fleet-123/
gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d
12:56:24,266 INFO || - [WebSocketInvoker] Thread-12 - StartGameSessionRequest:
gameSessionId=arn:aws:gamelift:local::gamesession/fleet-123/
gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d, fleetId=fleet-123, gameSessionName=null,
maxPlayers=4, properties=[], ipAddress=127.0.0.1, port=33430, gameSessionData?=false,
matchmakerData?=false, dnsName=localhost
12:56:24,564 INFO || - [CreateGameSessionDispatcher] Thread-12 - GameSession with
id: arn:aws:gamelift:local::gamesession/fleet-123/gsess-59f6cc44-4361-42f5-95b5-
fdb5825c0f3d created

```

```
12:56:24,585 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
GameLift.DescribeGameSessions
12:56:24,585 INFO || - [DescribeGameSessionsDispatcher] Thread-12 - Received API call
to describe game sessions with input: {"FleetId":"fleet-123"}
12:56:24,660 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
GameLift API to use: GameSessionActivate for pId 20728
12:56:24,661 INFO || - [GameSessionActivateHandler] GameLiftSdkHttpHandler-thread-0 -
Received API call for GameSessionActivate from 20728
12:56:24,678 INFO || - [GameSessionActivateHandler] GameLiftSdkHttpHandler-thread-0
- GameSessionActivate data: gameSessionId: "arn:aws:gamelift:local::gamesession/
fleet-123/gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d"
```

關閉伺服器程序

Note

本主題參考 Unity 1.0.0 版的 Amazon GameLift Servers 外掛程式，其使用伺服器 SDK 4.x 或更早版本。

完成範例遊戲後，請在 Unity 中關閉伺服器。

1. 在遊戲用戶端中，選擇結束或關閉視窗以停止遊戲用戶端。
2. 在 Unity 的本機測試視窗中，選擇停止或關閉遊戲伺服器視窗以停止伺服器。

使用 Amazon GameLift Servers 受管 EC2 託管的開發藍圖

此藍圖引導您如何為多玩家遊戲開發 Amazon GameLift Servers 受管 EC2 託管解決方案。Amazon GameLift Servers 提供多種遊戲託管選項；如需這些選項的詳細資訊，請參閱 [Amazon GameLift Servers 解決方案](#)。

透過 Amazon GameLift Servers 受管託管，您的遊戲伺服器會託管基於 AWS 雲端的虛擬運算資源，這些資源 Amazon GameLift Servers 擁有並根據您的組態運作。您可以取得 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體的安全性、可靠性和全域可用性，這些執行個體已進一步最佳化，可與多玩家遊戲託管搭配使用。透過自動伺服器部署、生命週期處理和容量自動擴展等工具，Amazon GameLift Servers 簡化託管管理。

Amazon GameLift Servers 受管解決方案由下列元件組成：

- 一或多個Amazon GameLift Servers受管機群，使用針對多玩家遊戲託管最佳化的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。
- 遊戲伺服器組建，與適用於 的伺服器 SDK 整合Amazon GameLift Servers，可跨所有機群部署。
- 與 AWS SDK 整合的遊戲用戶端和後端服務，可與服務互動Amazon GameLift Servers並請求遊戲工作階段。
- 將新遊戲工作階段與可用遊戲伺服器放在所有機群中的Amazon GameLift Servers佇列。
- (選用) FlexMatch配對建構器，可建立多玩家配對並為其設定遊戲工作階段。

此藍圖提供簡化的路徑，讓您的多玩家遊戲透過Amazon GameLift Servers受管 EC2 託管成功啟動和執行。準備好必要的元件後，您可以繼續反覆執行遊戲開發，並自訂您的託管解決方案。隨著您更接近啟動，請參閱這些[準備您的遊戲以使用 Amazon GameLift Servers 託管啟動](#)，以協助您準備生產層級用量的託管解決方案。

開始使用 Unreal Engine 和 Unity 的Amazon GameLift Servers外掛程式

若要加快部署速度，請嘗試 Unreal Engine 和 Unity 的[Amazon GameLift Servers外掛程式](#)。它提供引導式 UI 工作流程，以最少的設定快速部署遊戲伺服器，因此您可以嘗試執行中的遊戲元件。然後，您可以在此基礎上建置，為您的遊戲建立自訂託管解決方案。如需詳細資訊，請參閱[使用 Amazon GameLift Servers 外掛程式探索](#)。

步驟 1：準備您的遊戲伺服器以使用 Amazon GameLift Servers

將功能新增至您的遊戲伺服器，以便在部署服務進行託管時與Amazon GameLift Servers服務通訊。

- 為您的遊戲專案取得適用於 Amazon GameLift Servers(5.x 版) 的伺服器 SDK。伺服器 SDK 提供 C++、C# 和 Go 版本。 [下載Amazon GameLift Servers伺服器 SDK](#)。
- 修改遊戲伺服器程式碼以新增伺服器 SDK 功能。如需準則，請參閱[將遊戲與自訂遊戲伺服器整合](#)。至少，請執行下列動作：
 - 新增程式碼以初始化 Amazon GameLift Servers SDK，並建立與 Amazon GameLift Servers服務的 WebSocket 連線。使用伺服器 SDK 動作 `InitSdk()`。
 - 新增程式碼，以在伺服器程序準備好託管遊戲工作階段時向Amazon GameLift Servers服務報告。使用伺服器 SDK 動作 `ProcessReady()`。
 - 實作必要的回呼函數 `OnProcessTerminate()`、和 `OnStartGameSession()`。透過這些函數，遊戲伺服器程序可以維持與服務的連線Amazon GameLift Servers、在提示時啟動遊戲工作階段Amazon GameLift Servers，以及回應結束遊戲伺服器程序的提示。

- 新增程式碼，以在伺服器程序結束遊戲工作階段時向Amazon GameLift Servers服務報告。使用伺服器 SDK 動作 `ProcessEnding()`。
- 封裝遊戲伺服器組建。使用建置檔案、相依性和支援的軟體建立安裝指令碼。請參閱 [封裝遊戲建置檔案](#)。建議使用 Amazon Simple Storage Service (Amazon S3) 儲存貯體來存放遊戲組建的版本。
- 測試您的遊戲伺服器整合。針對此任務，我們建議為本機工作站設定 Amazon GameLift ServersAnywhere 機群，如中所述[使用 Amazon GameLift ServersAnywhere 設定本機測試](#)。在此步驟中，請在測試裝置上手動安裝遊戲伺服器組建，並啟動伺服器程序。使用 AWS CLI 請求新的遊戲工作階段，並確認Amazon GameLift Servers服務是否成功提示您的伺服器程序啟動遊戲工作階段。

步驟 2：準備您的遊戲用戶端以加入託管遊戲工作階段

為您的遊戲用戶端建立請求加入遊戲工作階段、取得連線資訊，然後直接連線至託管遊戲工作階段的方法。最常見的方法是設定後端服務功能，做為遊戲用戶端和 Amazon GameLift Servers 服務之間的中間人。此方法可保護您的託管資源，並讓您更能控制玩家如何進入遊戲工作階段。

- 建置後端服務功能以進行託管。後端服務會與服務通訊，Amazon GameLift Servers並將連線資訊交付至遊戲用戶端。此功能包括開始遊戲工作階段、將玩家放入遊戲中，以及擷取遊戲工作階段資訊。如需準則，請參閱[將遊戲與自訂遊戲伺服器整合](#)。至少，請執行下列動作：
- 取得適用於的 AWS 開發套件，Amazon GameLift Servers並將其新增至您的後端服務專案。請參閱[Amazon GameLift Servers用戶端服務的 SDK 資源](#)。
- 新增程式碼以初始化Amazon GameLift Servers用戶端和存放金鑰設定。請參閱 [在後端服務 Amazon GameLift Servers上設定](#)。
- 新增功能以呼叫 AWS SDK 動作`CreateGameSession()`，並將遊戲工作階段連線資訊提供給遊戲用戶端。請參閱[在特定機群上建立遊戲工作階段](#)。

呼叫`CreateGameSession()`是請求新遊戲工作階段的方便起點。在準備好遊戲工作階段放置系統之後（請參閱步驟 3），您會將此程式碼取代為呼叫 `StartGameSessionPlacement()`（或者`StartMatchmaking()`如果您使用的是)FlexMatch。

如需設計後端服務的指引，請參閱 [設計您的遊戲用戶端服務](#)。

- 將功能新增至遊戲用戶端，讓玩家加入託管遊戲工作階段。遊戲用戶端向後端服務提出請求，而不是直接向 Amazon GameLift Servers請求。在後端服務提供遊戲工作階段連線資訊後，遊戲用戶端會直接與遊戲工作階段連線，以玩遊戲。
- 測試您的遊戲用戶端整合。您可以搭配本機工作站使用相同的 Amazon GameLift ServersAnywhere 機群進行測試。

為了快速反覆開發或與多人團隊合作，我們建議您設定[雲端型測試環境](#)。此 Amazon GameLift Servers Toolkit 解決方案模擬 Amazon GameLift Servers 受管機群的行為，但可讓您以最短的周轉時間更新遊戲伺服器組建。

步驟 3：設定遊戲工作階段置放

自訂您希望 Amazon GameLift Servers 如何處理新遊戲工作階段的請求，並尋找可用的遊戲伺服器來託管它們。Amazon GameLift Servers 會自動追蹤所有機群上所有遊戲伺服器的可用性。當遊戲用戶端傳送加入遊戲工作階段的請求時，會根據一組定義的優先順序來 Amazon GameLift Servers 尋找「可能的最佳」配置，例如最低延遲、成本和可用性。

- 建立遊戲工作階段佇列，以使用可用的遊戲伺服器放置新的遊戲工作階段。佇列是遊戲工作階段放置的主要機制。如需準則，請參閱[建立遊戲工作階段佇列](#)。
- 至少，將 Anywhere 機群新增為佇列中的目的地。所有其他設定都是選用的自訂。
- 在您的後端服務程式碼中，將 `CreateGameSession()` 呼叫轉換為 `StartGameSessionPlacement()`。請參閱[在多個位置佇列中建立遊戲工作階段](#)。
- 建立機制，在遊戲工作階段準備好加入時通知遊戲用戶端。在開發過程中，您可以使用呼叫 `DescribeGameSessionPlacement` 來輪詢遊戲工作階段狀態。不過，在使用佇列處理大量之前，您需要啟用事件通知。請參閱[設定遊戲工作階段放置的事件通知](#)。
- （選用）新增 FlexMatch 配對元件。如需指引，請參閱[Amazon GameLift Servers FlexMatch 開發人員指南](#)。

步驟 4：建立受管雲端型機群

到目前為止，您已使用自我管理的 Anywhere 機群來測試和迭代遊戲元件，並且已微調遊戲工作階段放置。解決方案的最後一個部分是設定生產系統所需的託管資源類型。若要開始規劃和設定生產，您想要轉換為使用 Amazon GameLift Servers 受管機群。

- 封裝遊戲伺服器建置並上傳至 Amazon GameLift Servers。使用建置檔案、相依性和支援的軟體建立安裝指令碼。請參閱[部署用於 Amazon GameLift Servers 託管的自訂伺服器建置](#)。您可以使用主控台或 Amazon GameLift Servers CLI 將組建上傳至 AWS。

在上傳建置之前，請先決定 AWS 區域您要建立機群的內容。您必須將組建上傳到相同的區域。如需選擇機群位置的詳細資訊，請參閱[機群位置](#)。

- 建立受管 EC2 機群。當您建立機群時，會Amazon GameLift Servers立即開始部署遊戲伺服器組建以進行託管。您可以設定受管機群的許多層面。如需準則，請參閱[建立Amazon GameLift Servers受管 EC2 機群](#)。至少，請執行下列動作：
 - 為機群命名並指定要部署的上傳遊戲組建。
 - 選擇機群的隨需執行個體，然後選取機群位置中可用的執行個體類型。Spot 機群是寶貴的選項，但需要額外的設計和組態。
 - 建立機群的執行時間組態。至少，指定遊戲伺服器可執行檔的啟動路徑。
 - 指定連接埠設定，以允許傳入流量存取您的遊戲伺服器。
- 將受管機群新增至佇列。在您的遊戲工作階段佇列中，使用新增受管機群取代 Anywhere 機群。
- 使用受管機群測試遊戲託管。此時，您應該能夠測試整個託管週期，讓遊戲用戶端請求遊戲工作階段、取得連線資訊，以及成功連線至遊戲工作階段。

步驟 5：自訂您的受管機群

準備啟動遊戲時，您需要微調受管託管資源。要考慮的一些決策包括：

- 考慮新增 Spot 機群以節省成本。請參閱 [教學課程：使用 Spot 執行個體建立 Amazon GameLift Servers佇列](#)。
- 如果您的遊戲伺服器需要通訊其他 AWS 資源，請設定 IAM 角色來管理存取。請參閱 [與機群中的其他 AWS 資源通訊](#)。
- 決定要在地理上放置遊戲伺服器的位置。將遠端位置新增至受管機群。請參閱 [自訂您的 Amazon GameLift Servers EC2 受管機群](#)。
- 選取執行個體類型和大小，並將執行時間設定為執行多個伺服器程序，以最佳化機群效能。請參閱 [管理 如何Amazon GameLift Servers啟動遊戲伺服器](#)。
- 試用受管機群的遊戲工作階段置放選項，包括自訂優先順序設定。請參閱 [自訂遊戲工作階段佇列](#)。
- 設定自動容量擴展以滿足預期的玩家需求。請參閱 [使用 擴展遊戲託管容量 Amazon GameLift Servers](#)。
- 在其他 中設定待命機群 AWS 區域，並視需要修改佇列和自動擴展以處理容錯移轉。
- 設定託管可觀測性工具，包括分析和記錄。請參閱 [監控 Amazon GameLift Servers](#)。
- 使用[基礎設施即程式碼 \(IaC\)](#) 自動化部署。請參閱 [使用 管理Amazon GameLift Servers託管資源 AWS CloudFormation](#)。

Amazon GameLift Servers 支援針對任何部署特定的組態使用 AWS CloudFormation 範本。您也可以使用 AWS Cloud Development Kit (AWS CDK) 來定義您的 Amazon GameLift Servers 資源。如需的詳細資訊 AWS CDK，請參閱 [AWS Cloud Development Kit \(AWS CDK\) 開發人員指南](#)。

若要管理 AWS CloudFormation 堆疊的部署，建議您使用持續整合和持續交付 (CI/CD) 工具和服務，例如 AWS CodePipeline。這些工具可協助您在建置遊戲伺服器二進位檔時，自動部署 或獲得核准。使用 CI/CD 工具或服務時，新遊戲伺服器版本的資源部署如下所示：

- 建置和測試您的遊戲伺服器二進位檔。
- 將二進位檔案上傳至 Amazon GameLift Servers。
- 使用新建置部署新機群。
- 將新機群新增至您的遊戲工作階段佇列，並移除具有先前建置版本的機群。
- 當具有先前建置的機群不再託管作用中的遊戲工作階段時，請刪除這些機群的 AWS CloudFormation 堆疊。

使用Amazon GameLift Servers受管容器託管的開發藍圖

此藍圖會引導您如何為容器化遊戲伺服器開發Amazon GameLift Servers受管託管解決方案。受管容器只是提供的一種託管解決方案Amazon GameLift Servers。如需託管選項的詳細資訊，請參閱 [Amazon GameLift Servers 解決方案](#)。

使用的受管容器解決方案Amazon GameLift Servers具有下列元件：

- 一或多個容器機群，使用針對多玩家遊戲託管最佳化的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。
- 包含遊戲伺服器建置的容器映像，上傳至 Amazon Elastic Container Registry (Amazon ECR) 私有儲存庫。遊戲伺服器建置與適用於的伺服器 SDK 整合Amazon GameLift Servers，並建置在 Linux 上執行。
- 代表遊戲用戶端與 Amazon GameLift Servers服務互動的後端服務。後端服務使用適用於的服務 API 中的功能Amazon GameLift Servers，這是 SDK 的一部分 AWS。
- Amazon GameLift Servers 遊戲工作階段佇列，可處理新遊戲工作階段的請求、在所有機群中搜尋可用的遊戲伺服器，並提示遊戲伺服器啟動遊戲工作階段。
- (選用) FlexMatch配對建構器，可建立多玩家配對並為其設定遊戲工作階段。

此藍圖提供簡化的路徑，讓您的容器化遊戲伺服器能夠使用Amazon GameLift Servers受管容器成功啟動和執行。準備好必要的元件後，您可以繼續反覆執行遊戲開發，並自訂您的託管解決方案。隨著您更接近啟動，請參閱這些[準備您的遊戲以使用 Amazon GameLift Servers 託管啟動](#)以協助您準備生產層級用量的託管解決方案。

 使用這些受管容器的工具加速入門：

- [容器入門套件](#)可簡化整合和機群設定。它會為您的遊戲伺服器新增必要的遊戲工作階段管理功能，並使用預先設定的範本來建置容器機群和遊戲伺服器的自動化部署管道。部署之後，請使用 Amazon GameLift Servers主控台和 API 工具來監控機群效能、管理遊戲工作階段和分析指標。
- 對於 Unreal Engine 和 Unity 開發人員，請使用[Amazon GameLift Servers外掛程式](#)來整合遊戲伺服器，並從遊戲引擎的開發環境中建置容器機群。外掛程式的引導式工作流程可協助您建立快速、簡單的解決方案，並使用受管容器進行雲端託管。您可以建置此基礎，為您的遊戲建立自訂託管解決方案。

步驟 1：準備您的遊戲伺服器以使用 Amazon GameLift Servers

將功能新增至您的遊戲伺服器，以便在部署服務進行託管時與Amazon GameLift Servers服務通訊。

- 為您的遊戲專案取得適用於 Amazon GameLift Servers(5.2 版或更新版本) 的伺服器 SDK。伺服器 SDK 提供 C++、C# 和 Go 版本。 [下載適用於 的伺服器 SDKAmazon GameLift Servers](#)。伺服器 SDK 提供 C++、C# 和 Go 版本。
- 修改遊戲伺服器程式碼以新增伺服器 SDK 功能。如需準則，請參閱[將遊戲與自訂遊戲伺服器整合](#)。至少，請執行下列動作：
 - 新增程式碼以初始化 Amazon GameLift Servers SDK，並建立與 Amazon GameLift Servers服務的 WebSocket 連線。使用伺服器 SDK 動作 `InitSdk()`。
 - 新增程式碼，以在伺服器程序準備好託管遊戲工作階段時向Amazon GameLift Servers服務報告。使用伺服器 SDK 動作 `ProcessReady()`。
 - 實作必要的回呼函數 `OnStartGameSession()`、和 `OnProcessTerminate()`。透過這些函數，遊戲伺服器程序可以維持與服務的連線Amazon GameLift Servers、在提示時啟動遊戲工作階段Amazon GameLift Servers，以及回應結束遊戲伺服器程序的請求。
 - 新增程式碼，以在伺服器程序結束遊戲工作階段時向Amazon GameLift Servers服務報告。使用伺服器 SDK 動作 `ProcessEnding()`。

- 封裝遊戲伺服器組建。建置遊戲伺服器以在 Linux 上執行。準備執行遊戲伺服器所需的組建和其他檔案。如果您是在 Windows 上進行開發，此步驟可能涉及設定個別的 Linux 工作區，或使用 Windows 子系統 for Linux (WSL) 等工具。您需要 Linux 環境來測試遊戲伺服器建置，以及建置和測試容器映像。
- 測試您的遊戲伺服器整合。確認您的整合遊戲伺服器可以連線至 Amazon GameLift Servers 服務，並回應提示。建議您設定簡單的 Amazon GameLift Servers Anywhere 機群，並將本機工作站做為測試主機，如中所述[使用 Amazon GameLift Servers Anywhere 設定本機測試](#)。在測試主機上安裝遊戲伺服器建置，並啟動伺服器程序。使用 AWS CLI 請求新的遊戲工作階段，並確認 Amazon GameLift Servers 服務是否成功提示您的伺服器程序啟動遊戲工作階段。

步驟 2：準備您的遊戲用戶端以加入託管遊戲工作階段

為您的遊戲用戶端建立請求加入遊戲工作階段、取得連線資訊，然後直接連線至託管遊戲工作階段的方法。最常見的方法是設定後端服務功能，做為遊戲用戶端和 Amazon GameLift Servers 服務之間的中間人。此方法可保護您的託管資源，並讓您更能控制玩家如何進入遊戲工作階段。

- 建置後端服務功能以進行託管。後端服務會與服務通訊，Amazon GameLift Servers 並將連線資訊交付至遊戲用戶端。此功能包括開始遊戲工作階段、將玩家放入遊戲中，以及擷取遊戲工作階段資訊。如需準則，請參閱[將遊戲與自訂遊戲伺服器整合](#)。至少，請執行下列動作：
- 取得適用於的 AWS 開發套件，Amazon GameLift Servers 並將其新增至您的後端服務專案。請參閱[Amazon GameLift Servers 用戶端服務的 SDK 資源](#)。
- 新增程式碼以初始化 Amazon GameLift Servers 用戶端和存放金鑰設定。請參閱 [在後端服務 Amazon GameLift Servers 上設定](#)。
- 新增功能以呼叫 AWS SDK 動作 `CreateGameSession()`，並將遊戲工作階段連線資訊提供給遊戲用戶端。請參閱[在特定機群上建立遊戲工作階段](#)。

呼叫 `CreateGameSession()` 是請求新遊戲工作階段的方便起點。在準備好遊戲工作階段放置系統之後（請參閱步驟 3），您會將此程式碼取代為呼叫 `StartGameSessionPlacement()`（或者 `StartMatchmaking()` 如果您使用的是 `FlexMatch`）。

如需設計後端服務的指引，請參閱 [設計您的遊戲用戶端服務](#)。

- 將功能新增至遊戲用戶端，讓玩家加入託管遊戲工作階段。遊戲用戶端會向您的後端服務提出請求，而不是直接向 Amazon GameLift Servers 提出請求。後端服務提供遊戲工作階段連線資訊後，遊戲用戶端會直接與遊戲工作階段連線，以玩遊戲。

- 測試您的遊戲用戶端整合。您可以使用現有的 Amazon GameLift ServersAnywhere 機群搭配本機工作站進行測試。使用新的後端服務請求新的遊戲工作階段，並確認：(1) Amazon GameLift Servers 服務成功提示您的伺服器程序啟動遊戲工作階段，以及 (2) 遊戲用戶端可以連線至遊戲工作階段。

步驟 3：設定遊戲工作階段放置

自訂您希望 Amazon GameLift Servers 如何處理新遊戲工作階段的請求，並尋找可用的遊戲伺服器來託管它們。Amazon GameLift Servers 會自動追蹤所有機群上所有遊戲伺服器的可用性。當遊戲用戶端傳送加入遊戲工作階段的請求時，會根據一組定義的優先順序 Amazon GameLift Servers 尋找「可能的最佳」放置，例如最低延遲、成本和可用性。

- 建立遊戲工作階段佇列，以使用可用的遊戲伺服器放置新的遊戲工作階段。佇列是遊戲工作階段放置的主要機制。如需準則，請參閱[建立遊戲工作階段佇列](#)。
 - 至少，將 Anywhere 機群新增為佇列中的目的地。所有其他設定都是選用的自訂。
- 在您的後端服務程式碼中，將 `CreateGameSession()` 呼叫轉換為 `StartGameSessionPlacement()`。請參閱[在多個位置佇列中建立遊戲工作階段](#)。
- 建立機制，在遊戲工作階段準備好加入時通知遊戲用戶端。在開發過程中，您可以使用呼叫來輪詢遊戲工作階段狀態 `DescribeGameSessionPlacement`。不過，在使用佇列處理大量之前，您需要啟用事件通知。請參閱[設定遊戲工作階段放置的事件通知](#)。
- 新增 FlexMatch 配對（選用）。建置配對規則集並建立配對組態，以使用您的遊戲工作階段佇列。如需設定配對系統的指引，請參閱[Amazon GameLift Servers FlexMatch 開發人員指南](#)。
- 測試置放系統。您可以使用現有的 Amazon GameLift ServersAnywhere 機群搭配本機工作站進行測試。使用後端服務請求新的遊戲工作階段，並確認該 Amazon GameLift Servers 服務已成功提示您的伺服器程序啟動遊戲工作階段。

步驟 4：建立遊戲伺服器容器映像

成功整合遊戲伺服器後，請使用遊戲伺服器可執行檔建立容器映像。將它存放在 Amazon Elastic Container Registry (Amazon ECR) 儲存庫中，以便與搭配使用 Amazon GameLift Servers。如需詳細說明，請參閱[建置的容器映像 Amazon GameLift Servers](#)。

- 取得遊戲伺服器容器的 Dockerfile 範本（由提供 Amazon GameLift Servers）。修改遊戲伺服器建置檔案的檔案。
- 建置遊戲伺服器容器映像。在 Linux 環境中使用 Docker 工具來建立映像。
- 將您的容器映像推送至 Amazon ECR。使用 AWS 區域您計劃部署容器機群的相同 AWS 帳戶和，在 Amazon ECR 中建立公有或私有儲存庫。將容器映像推送至其中。

- 使用 Anywhere 機群測試容器映像（選用）。在將容器映像部署到雲端託管的容器機群之前，建議您先在本機測試容器映像。您可以使用現有的 Amazon GameLift Servers Anywhere 機群搭配本機工作站進行測試。安裝並執行遊戲伺服器容器，並確認：(1) Amazon GameLift Servers 服務成功提示您的伺服器程序啟動遊戲工作階段，以及 (2) 遊戲用戶端可以連線至遊戲工作階段。

步驟 5：建立雲端容器機群

到目前為止，您已使用自我管理的 Anywhere 機群來測試和迭代遊戲元件。解決方案的最後一個部分是設定生產系統所需的雲端型託管資源。若要開始規劃和設定生產環境，您要設定 Amazon GameLift Servers 受管容器機群，並自訂用於生產環境。

- 建立容器群組定義。容器群組定義描述機群的容器架構。並識別要部署的容器映像。請參閱 [建立容器機群的 Amazon GameLift Servers 容器群組定義](#)。在儲存容器映像的相同 AWS 區域中建立容器群組定義。如需選擇機群位置的詳細資訊，請參閱 [機群位置](#)。至少，請執行下列動作：
 - 建立遊戲伺服器容器群組定義。
 - 使用遊戲伺服器建置新增具有容器映像的容器定義。
 - 設定容器遊戲伺服器程序的連接埠範圍。
- 建立受管容器機群。當您建立機群時，會 Amazon GameLift Servers 立即開始部署遊戲伺服器組建以進行託管。您可以設定受管機群的許多層面。如需準則，請參閱 [建立 Amazon GameLift Servers 受管容器機群](#)。至少，請執行下列動作：
 - 設定容器機群的 AWS Identity and Access Management (IAM) 服務角色。請參閱 [設定的 IAM 服務角色 Amazon GameLift Servers](#)。
 - 指定要部署到機群執行個體的遊戲伺服器容器群組定義。
 - 使用所有其他參數可用的預設值。會 Amazon GameLift Servers 計算一些參數以獲得最佳組態。
- 將容器機群新增至佇列。在您的遊戲工作階段佇列中，將 Anywhere 測試機群取代為您的受管容器機群。
- 使用容器機群測試遊戲託管。此時，您應該能夠測試整個解決方案。啟動遊戲用戶端，並透過後端服務請求遊戲工作階段。取得連線資訊並連線至容器機群上的遊戲工作階段。
- 在您的機群部署上反覆執行。您可以更新容器群組定義和機群組態，然後將更新部署到現有的機群。

步驟 6：自訂您的受管容器解決方案

準備啟動遊戲時，您需要微調受管託管資源。要考慮的一些決策包括：

- 最佳化您的容器機群組態。請參閱 [自訂 Amazon GameLift Servers 容器機群](#)。

- 考慮新增 Spot 機群以節省成本。請參閱 [教學課程：使用 Spot 執行個體建立 Amazon GameLift Servers 佇列](#)。
- 如果您的遊戲伺服器需要通訊其他 AWS 資源，請設定 IAM 角色來管理存取。請參閱 [與機群中的其他 AWS 資源通訊](#)。
- 決定要在地理上放置遊戲伺服器的位置。將遠端位置新增至受管機群。請參閱 [自訂您的 Amazon GameLift Servers EC2 受管機群](#)。
- 試用受管機群的遊戲工作階段置放選項，包括自訂優先順序設定。請參閱 [自訂遊戲工作階段佇列](#)。
- 設定自動容量擴展以滿足預期的玩家需求。請參閱 [使用 擴展遊戲託管容量 Amazon GameLift Servers](#)。
- 在其他 中建立機群 AWS 區域，並視需要修改佇列和自動擴展以處理容錯移轉。
- 設定託管可觀測性工具，包括分析和記錄。請參閱 [監控 Amazon GameLift Servers](#)。
- 使用[基礎設施即程式碼 \(IaC\)](#) 自動化機群部署。請參閱 [使用 管理 Amazon GameLift Servers 託管資源 AWS CloudFormation](#)。

Amazon GameLift Servers 支援針對任何部署特定的組態使用 AWS CloudFormation 範本。您也可以使用 AWS Cloud Development Kit (AWS CDK) 來定義您的 Amazon GameLift Servers 資源。如需的詳細資訊 AWS CDK，請參閱 [AWS Cloud Development Kit \(AWS CDK\) 開發人員指南](#)。

若要管理 AWS CloudFormation 堆疊的部署，建議您使用持續整合和持續交付 (CI/CD) 工具和服務，例如 AWS CodePipeline。這些工具可協助您在每次建置遊戲伺服器二進位檔時，自動部署或獲得核准。使用 CI/CD 工具或服務時，新遊戲伺服器版本的資源部署如下所示：

- 建置和測試您的遊戲伺服器二進位檔。
- 將二進位檔上傳至 Amazon GameLift Servers。
- 使用新建置部署新機群。
- 將新機群新增至您的遊戲工作階段佇列，並移除具有先前建置版本的機群。
- 當具有先前建置的機群不再託管作用中遊戲工作階段時，請刪除這些機群的 AWS CloudFormation 堆疊。

使用 Amazon GameLift Servers Anywhere 託管的開發藍圖

此藍圖會引導您如何為多玩家遊戲開發託管解決方案，以搭配您自己的資源（內部部署硬體或虛擬機器）使用。Amazon GameLift Servers 提供多種遊戲託管選項；如需這些選項的詳細資訊，請參閱 [Amazon GameLift Servers 解決方案](#)。

透過 Amazon GameLift Servers Anywhere 託管，您的遊戲伺服器託管在您提供和管理的運算資源上。您可以建立具有所需組態的 Anywhere 機群，且無論玩家位於何處，位置都在地理上。為 Anywhere 機群 Amazon GameLift Servers 提供下列功能：

- 根據您的組態為您處理遊戲工作階段置放程序，包括：
 - 追蹤您 Anywhere 機群的遊戲伺服器可用性。
 - 處理來自遊戲用戶端服務的遊戲請求，以及將遊戲請求與可用伺服器配對。
 - 提示 Anywhere 機群上的遊戲伺服器啟動遊戲工作階段。
 - 將連線詳細資訊傳遞回遊戲用戶端。
- 收集工作階段置放程序的效能指標，以及遊戲工作階段和玩家的使用指標。
- 支援完整的 FlexMatch 配對功能集，因此您可以建置配對建構器，並將其與遊戲工作階段放置系統整合。
- 提供 Amazon GameLift Servers 代理程式來處理 Anywhere 機群上的金鑰主機管理任務。
- 支援結合 Amazon GameLift Servers 受管機群，提供靈活的混合解決方案。

Amazon GameLift Servers Anywhere 解決方案由下列元件組成：

- 一或多個具有內部部署或其他託管資源的 Amazon GameLift Servers Anywhere 機群，使用您現有的組態管理和部署工具進行管理。（您可以選擇使用 AWS Systems Manager。）
- 遊戲伺服器組建，與適用於的伺服器 SDK 整合 Amazon GameLift Servers，可跨所有機群部署。
- 與 AWS SDK 整合的遊戲用戶端和後端服務，可與服務互動 Amazon GameLift Servers 並請求遊戲工作階段。
- 將新遊戲工作階段與可用遊戲伺服器放在所有機群中的 Amazon GameLift Servers 佇列。
- （選用）FlexMatch 配對建構器，可建立多玩家配對並為其設定遊戲工作階段。

此藍圖提供簡化的路徑，讓您的多玩家遊戲在 Amazon GameLift Servers Anywhere 託管中成功啟動和執行。準備好必要的元件後，您可以繼續反覆執行遊戲開發，並自訂您的託管解決方案。隨著您更接近啟動，請參閱這些[準備您的遊戲以使用 Amazon GameLift Servers 託管啟動](#)，以協助您準備生產層級用量的託管解決方案。

開始使用 Unreal Engine 和 Unity 的 Amazon GameLift Servers 外掛程式

若要加快部署速度，請嘗試 Unreal Engine 和 Unity 的[Amazon GameLift Servers 外掛程式](#)。它提供引導式 UI 工作流程，以最少的設定快速部署遊戲伺服器，因此您可以嘗試執行中的遊戲

元件。然後，您可以在此基礎上建置，為您的遊戲建立自訂託管解決方案。如需詳細資訊，請參閱[使用 Amazon GameLift Servers 外掛程式探索](#)。

步驟 1：準備您的遊戲伺服器以使用 Amazon GameLift Servers

將功能新增至您的遊戲伺服器，以便在部署服務進行託管時與Amazon GameLift Servers服務通訊。

- 為您的遊戲專案取得適用於 Amazon GameLift Servers(5.x 版) 的伺服器 SDK。伺服器 SDK 提供 C++、C# 和 Go 版本。 [下載Amazon GameLift Servers伺服器 SDK](#)。
- 修改遊戲伺服器程式碼以新增伺服器 SDK 功能。如需準則，請參閱[將遊戲與自訂遊戲伺服器整合](#)。至少，請執行下列動作：
 - 新增程式碼以初始化 Amazon GameLift Servers SDK，並建立與 Amazon GameLift Servers服務的 WebSocket 連線。使用伺服器 SDK 動作InitSdk()，並包含 Anywhere 機群所需的伺服器參數。
 - 新增程式碼，以在伺服器程序準備好託管遊戲工作階段時向Amazon GameLift Servers服務報告。使用伺服器 SDK 動作 ProcessReady()。
 - 實作必要的回呼函數 OnProcessTerminate()、和 OnStartGameSession()。透過這些函數，遊戲伺服器程序可以維持與服務的連線Amazon GameLift Servers、在提示時啟動遊戲工作階段Amazon GameLift Servers，以及回應結束遊戲伺服器程序的提示。
 - 新增程式碼，以在伺服器程序結束遊戲工作階段時向Amazon GameLift Servers服務報告。使用伺服器 SDK 動作 ProcessEnding()。
- 封裝遊戲伺服器組建。使用建置檔案、相依性和支援的軟體建立安裝指令碼。請參閱 [封裝遊戲建置檔案](#)。建議使用 Amazon Simple Storage Service (Amazon S3) 儲存貯體來存放遊戲組建的版本。
- 測試您的遊戲伺服器整合。針對此任務，我們建議為本機工作站設定 Amazon GameLift ServersAnywhere 機群，如中所述[使用 Amazon GameLift ServersAnywhere 設定本機測試](#)。在此步驟中，請在測試裝置上手動安裝遊戲伺服器組建，並啟動伺服器程序。使用 AWS CLI 請求新的遊戲工作階段，並確認Amazon GameLift Servers服務是否成功提示您的伺服器程序啟動遊戲工作階段。

步驟 2：準備您的遊戲用戶端以加入託管遊戲工作階段

為您的遊戲用戶端建立請求加入遊戲工作階段、取得連線資訊，然後直接連線至託管遊戲工作階段的方法。最常見的方法是設定後端服務功能，做為遊戲用戶端和 Amazon GameLift Servers 服務之間的中間人。此方法可保護您的託管資源，並讓您更能控制玩家如何進入遊戲工作階段。

- 建置後端服務功能以進行託管。後端服務會與服務通訊，Amazon GameLift Servers並將連線資訊交付至遊戲用戶端。此功能包括開始遊戲工作階段、將玩家放入遊戲中，以及擷取遊戲工作階段資訊。如需準則，請參閱[將遊戲與自訂遊戲伺服器整合](#)。至少，請執行下列動作：
- 取得適用於的 AWS 開發套件，Amazon GameLift Servers並將其新增至您的後端服務專案。請參閱[Amazon GameLift Servers用戶端服務的 SDK 資源](#)。
- 新增程式碼以初始化Amazon GameLift Servers用戶端和存放金鑰設定。請參閱 [在後端服務 Amazon GameLift Servers上設定](#)。
- 新增功能以呼叫 AWS SDK 動作CreateGameSession()，並將遊戲工作階段連線資訊提供給遊戲用戶端。請參閱[在特定機群上建立遊戲工作階段](#)。

呼叫CreateGameSession()是請求新遊戲工作階段的方便起點。在準備好遊戲工作階段放置系統之後（請參閱步驟 3），您會將此程式碼取代為呼叫 StartGameSessionPlacement()（或者StartMatchmaking()如果您使用的是)FlexMatch。

如需設計後端服務的指引，請參閱 [設計您的遊戲用戶端服務](#)。

- 將功能新增至遊戲用戶端，讓玩家加入託管遊戲工作階段。遊戲用戶端向後端服務提出請求，而不是直接向 Amazon GameLift Servers提出請求。在後端服務提供遊戲工作階段連線資訊後，遊戲用戶端會直接與遊戲工作階段連線，以玩遊戲。
- 測試您的遊戲用戶端整合。您可以搭配本機工作站使用相同的 Amazon GameLift ServersAnywhere 機群進行測試。

步驟 3：設定遊戲工作階段置放

自訂您希望Amazon GameLift Servers如何處理新遊戲工作階段的請求，並尋找可用的遊戲伺服器來託管它們。Amazon GameLift Servers會自動追蹤所有機群上所有遊戲伺服器的可用性。當遊戲用戶端傳送加入遊戲工作階段的請求時，會根據一組定義的優先順序來Amazon GameLift Servers尋找「可能的最佳」配置，例如最低延遲、成本和可用性。

- 建立遊戲工作階段佇列，以使用可用的遊戲伺服器放置新的遊戲工作階段。佇列是遊戲工作階段放置的主要機制。如需準則，請參閱[建立遊戲工作階段佇列](#)。
- 至少，將 Anywhere 機群新增為佇列中的目的地。所有其他設定都是選用的。
- 在您的後端服務程式碼中，將**CreateGameSession()**呼叫轉換為**StartGameSessionPlacement()**。請參閱[在多個位置佇列中建立遊戲工作階段](#)。
- 建立機制，在遊戲工作階段準備好加入時通知遊戲用戶端。在開發過程中，您可以使用呼叫 DescribeGameSessionPlacement 來輪詢遊戲工作階段狀態。不過，在使用佇列處理大量之前，您需要啟用事件通知。請參閱 [設定遊戲工作階段放置的事件通知](#)。

- (選用) 新增FlexMatch配對元件。如需指引，請參閱 [Amazon GameLift Servers FlexMatch 開發人員指南](#)。

步驟 4：使用 Amazon GameLift Servers 代理程式設定 Anywhere 機群

到目前為止，您一直使用本機裝置（註冊為 Anywhere 機群運算）來測試和迭代遊戲元件。下一個步驟是設定生產系統所需的機群類型。對於這些資源，請使用 Amazon GameLift Servers 代理程式來管理一些關鍵的運算中主機管理任務。如需詳細資訊，請參閱 [使用 Amazon GameLift Servers 代理程式](#)。

- 取得 Amazon GameLift Servers 代理程式，並將其新增至遊戲伺服器安裝套件。取得並建置 Agent [Amazon GameLift Servers Github 儲存庫中可用的 Agent](#) 原始程式碼。將產生的 JAR 檔案可執行檔放在與遊戲組建可執行檔相同的目錄中。
- 視需要修改 代理程式的啟動指令碼。確保在運算開始執行後立即啟動代理程式可執行檔。如需在託管運算上安裝和執行 代理程式的說明，請參閱 代理程式儲存庫中的讀我檔案。您的啟動命令應至少包含指定 Anywhere 機群 ID 和 AWS 區域、自訂位置和運算名稱的選項。

代理程式會自動為您處理下列任務，因此如果您一直使用指令碼處理這些任務，您可以移除它們：

- 呼叫 RegisterCompute() 將運算新增至 Anywhere 機群。
- 當遊戲伺服器連線到 Amazon GameLift Servers 服務時，呼叫 GetComputeAuthToken() 來驗證遊戲伺服器。代理程式會管理取得和重新整理身分驗證權杖，以供在運算上執行的所有遊戲伺服器程序使用。
- 根據一組執行時間指示，在運算上啟動新的伺服器程序。
- 為 Anywhere 機群中的運算建立執行期組態。您可以使用 Amazon GameLift Servers 主控台或 AWS CLI 來建立或修改機群的執行時間指示。代理程式會執行這些指示，並定期向 Amazon GameLift Servers 服務請求更新。
- 視需要設定或修改遊戲工作階段佇列。建立新的佇列（或更新現有的佇列），以使用隨 Amazon GameLift Servers 代理程式部署的 Anywhere 機群。
- 測試 代理程式與 Anywhere 機群的整合。檢查 代理程式是否根據執行時間組態正確啟動伺服器程序。

步驟 5：自訂您的 Anywhere 機群

準備啟動遊戲時，您需要微調受管託管資源。要考慮的一些決策包括：

- 視需要自動化啟動和關閉運算的程序，包括安裝和執行遊戲伺服器軟體。回收運算有助於確保定期更新，而且關閉運算可以在不需要時節省成本。
- 如果您的遊戲伺服器需要通訊其他 AWS 資源，請設定 IAM 角色來管理存取。請參閱 [與機群中的其他 AWS 資源通訊](#)。
- 決定要在地理上放置遊戲伺服器的位置。將遠端位置新增至受管機群。請參閱 [自訂您的 Amazon GameLift Servers EC2 受管機群](#)。
- 透過選取運算資源組態來最佳化機群效能，然後設定執行時間指示，以針對每個運算執行最佳數量的伺服器程序。
- 試用受管機群的遊戲工作階段置放選項，包括自訂優先順序設定。請參閱 [自訂遊戲工作階段佇列](#)。
- 建立機制來處理手動或自動化容量擴展，以滿足預期的玩家需求。考慮哪些因素應該提示系統增加或減少可用於託管遊戲工作階段的運算數量。
- 視需要設計和實作容錯移轉至其他資源。
- 設定託管可觀測性工具，包括分析和記錄。請參閱[監控 Amazon GameLift Servers](#)。

使用 混合託管的開發藍圖 Amazon GameLift Servers

此藍圖會引導您如何為多玩家遊戲開發託管解決方案。Amazon GameLift Servers提供多種遊戲託管選項；如需這些選項的詳細資訊，請參閱 [Amazon GameLift Servers 解決方案](#)。

混合解決方案使用託管資源的組合，包括由 管理的雲端型資源Amazon GameLift Servers和您自己的自我管理託管資源。如需混合託管的更詳細討論，請參閱以下文章：[使用 Amazon GameLift Servers Anywhere 託管混合遊戲伺服器](#)。使用 Amazon GameLift Servers，您可以設定使用常見元件和程序的混合解決方案，以便集中管理全域機群，並在所有類型的資源之間輕鬆移動負載。

混合架構包含下列元件：

- 一或多個Amazon GameLift Servers受管機群，使用針對多玩家遊戲託管最佳化的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。
- 一或多個 Amazon GameLift Servers Anywhere 機群，使用您現有的內部部署或其他託管資源，包括您的組態管理和部署工具。（您可以選擇使用 AWS Systems Manager。）
- 與適用於 的伺服器 SDK 整合的單一遊戲伺服器建置Amazon GameLift Servers，可跨所有機群部署。
- 與 SDK 整合的 AWS 單一遊戲用戶端和後端服務，可與服務互動Amazon GameLift Servers並請求遊戲工作階段。

- 共用Amazon GameLift Servers佇列，可放置具有可用遊戲伺服器器的新遊戲工作階段，並平衡所有機群的負載。
- 隨 Anywhere 機群一起部署的 Amazon GameLift Servers 代理程式，可簡化所有機群中運算的伺服器程序管理任務。
- (選用) FlexMatch配對建構器，可建立多玩家配對並為其設定遊戲工作階段。

此路線圖提供簡化的路徑，讓您的多玩家遊戲在搭配 的混合式託管解決方案中成功啟動和執行Amazon GameLift Servers。準備好必要的元件後，您可以繼續反覆執行遊戲開發，並自訂您的託管解決方案。隨著您更接近啟動，請參閱這些[準備您的遊戲以使用 Amazon GameLift Servers 託管啟動](#)，以協助您準備生產層級用量的託管解決方案。

開始使用 Amazon GameLift Servers 外掛程式

如果您使用 Unreal Engine 或 Unity 開發專案，請開始設定遊戲以使用Amazon GameLift Servers外掛程式託管。使用 外掛程式，您可以將Amazon GameLift ServersSDKs新增至您的遊戲專案，並使用引導式工作流程，透過 Anywhere 機群和 Amazon GameLift Servers受管機群，建置簡單的混合式託管解決方案工作版本。然後，您可以使用這些基礎知識來建置和視需要自訂。

步驟 1：準備您的遊戲伺服器以使用 Amazon GameLift Servers

將功能新增至您的遊戲伺服器，以便在部署服務進行託管時與Amazon GameLift Servers服務通訊。如果遊戲伺服器在Amazon GameLift Servers受管機群或 Anywhere 機群上執行，則需要相同的功能。

- 為您的遊戲專案取得適用於 Amazon GameLift Servers(5.x 版) 的伺服器 SDK。伺服器 SDK 提供 C++、C# 和 Go 版本。[下載適用於 的伺服器 SDKAmazon GameLift Servers](#)。
- 修改遊戲伺服器程式碼以新增伺服器 SDK 功能。如需準則，請參閱[將遊戲與自訂遊戲伺服器整合](#)。至少，請執行下列動作：
 - 新增程式碼以初始化 Amazon GameLift Servers SDK，並建立與 Amazon GameLift Servers服務的 WebSocket 連線。使用伺服器 SDK 動作 `InitSdk()`。在 Anywhere 機群運算上執行時，包含指定伺服器參數的程式碼。
 - 新增程式碼，以在伺服器程序準備好託管遊戲工作階段時向Amazon GameLift Servers服務報告。使用伺服器 SDK 動作 `ProcessReady()`。

- 實作必要的回呼函數 `OnProcessTerminate()`、和 `OnStartGameSession()`。透過這些函數，遊戲伺服器程序可以維持與服務的連線Amazon GameLift Servers、在提示時啟動遊戲工作階段Amazon GameLift Servers，以及回應結束遊戲伺服器程序的提示。
- 新增程式碼，以在伺服器程序結束遊戲工作階段時向Amazon GameLift Servers服務報告。使用伺服器 SDK 動作 `ProcessEnding()`。
- 封裝遊戲伺服器組建。使用建置檔案、相依性和支援的軟體建立安裝指令碼。請參閱 [封裝遊戲建置檔案](#)。建議使用 Amazon Simple Storage Service (Amazon S3) 儲存貯體來存放遊戲組建的版本。
- 測試您的遊戲伺服器整合。對於此任務，我們建議您使用本機工作站設定 Amazon GameLift ServersAnywhere 機群，如中所述[使用 Amazon GameLift ServersAnywhere 設定本機測試](#)。在此步驟中，請在測試裝置上手動安裝遊戲伺服器組建，並啟動伺服器程序。使用 AWS CLI 請求新的遊戲工作階段，並確認Amazon GameLift Servers服務是否成功提示您的伺服器程序啟動遊戲工作階段。

步驟 2：準備您的遊戲用戶端以加入託管遊戲工作階段

為您的遊戲用戶端建立請求加入遊戲工作階段、取得連線資訊，然後直接連線至託管遊戲工作階段的方法。最常見的方法是設定後端服務功能，做為遊戲用戶端和 Amazon GameLift Servers 服務之間的中間人。此方法可保護您的託管資源，並讓您更能控制玩家如何進入遊戲工作階段。

- 建置後端服務功能以進行託管。後端服務會與服務通訊，Amazon GameLift Servers並將連線資訊交付至遊戲用戶端。此功能包括開始遊戲工作階段、將玩家放入遊戲中，以及擷取遊戲工作階段資訊。如需準則，請參閱[將遊戲與自訂遊戲伺服器整合](#)。至少，請執行下列動作：
- 取得適用於的 AWS 開發套件，Amazon GameLift Servers並將其新增至您的後端服務專案。請參閱[Amazon GameLift Servers用戶端服務的 SDK 資源](#)。
- 新增程式碼以初始化Amazon GameLift Servers用戶端和存放金鑰設定。請參閱 [在後端服務 Amazon GameLift Servers上設定](#)。
- 新增功能以呼叫 AWS SDK 動作`CreateGameSession()`，並將遊戲工作階段連線資訊提供給遊戲用戶端。請參閱[在特定機群上建立遊戲工作階段](#)。

呼叫`CreateGameSession()`是請求新遊戲工作階段的方便起點。在準備好遊戲工作階段放置系統之後（請參閱步驟 3），您會將此程式碼取代為呼叫 `StartGameSessionPlacement()`（或者`StartMatchmaking()`如果您使用的是 `FlexMatch`。

如需設計後端服務的指引，請參閱 [設計您的遊戲用戶端服務](#)。

- 將功能新增至遊戲用戶端，讓玩家加入託管遊戲工作階段。遊戲用戶端向後端服務提出請求，而不是直接向 Amazon GameLift Servers 提出請求。在後端服務提供遊戲工作階段連線資訊後，遊戲用戶端會直接與遊戲工作階段連線，以玩遊戲。

- 測試您的遊戲用戶端整合。您可以搭配本機工作站使用相同的 Amazon GameLift ServersAnywhere 機群進行測試。

在開發階段，如果您想要測試遊戲組建在Amazon GameLift Servers受管機群中的行為，我們建議您也設定[雲端型測試環境](#)。此Amazon GameLift Servers工具組解決方案模擬受管機群的行為，但可讓您以最短的周轉時間更新遊戲伺服器組建。

步驟 3：設定遊戲工作階段置放

自訂您希望Amazon GameLift Servers如何處理新遊戲工作階段的請求，並尋找可用的遊戲伺服器來託管它們。Amazon GameLift Servers會自動追蹤所有機群上所有遊戲伺服器的可用性。當遊戲用戶端傳送加入遊戲工作階段的請求時，會根據一組定義的優先順序來Amazon GameLift Servers尋找「可能的最佳」配置，例如最低延遲、成本和可用性。

- 建立遊戲工作階段佇列，以使用可用的遊戲伺服器放置新的遊戲工作階段。佇列是遊戲工作階段放置的主要機制。如需準則，請參閱[建立遊戲工作階段佇列](#)。
 - 至少，將 Anywhere 機群新增為佇列中的目的地。所有其他設定都是選用的自訂。
- 在您的後端服務程式碼中，將**CreateGameSession()**呼叫轉換為**StartGameSessionPlacement()**。請參閱[在多位置佇列中建立遊戲工作階段](#)。
- 建立機制，在遊戲工作階段準備好加入時通知遊戲用戶端。在開發過程中，您可以使用呼叫DescribeGameSessionPlacement 來輪詢遊戲工作階段狀態。不過，在使用佇列處理大量之前，您需要啟用事件通知。請參閱 [設定遊戲工作階段放置的事件通知](#)。
- （選用）新增FlexMatch配對元件。如需指引，請參閱 [Amazon GameLift ServersFlexMatch開發人員指南](#)。

步驟 4：使用 Amazon GameLift Servers 代理程式設定 Anywhere 機群

到目前為止，您一直使用本機裝置（註冊為 Anywhere 機群運算）來測試和迭代遊戲元件。下一個步驟是設定生產系統所需的機群類型。從 Anywhere 機群開始，並新增 Amazon GameLift Servers 代理程式來管理一些關鍵的運算上主機管理任務。如需詳細資訊，請參閱[使用 Amazon GameLift Servers 代理程式](#)。

- 取得 Amazon GameLift Servers 代理程式，並將其新增至遊戲伺服器安裝套件。取得並建置 Agent [Amazon GameLift Servers Github 儲存庫中可用的 Agent](#) 原始程式碼。將產生的 JAR 檔案可執行檔放在與遊戲組建可執行檔相同的目錄中。

- 視需要修改 代理程式的啟動指令碼。確保在運算開始執行後立即啟動代理程式可執行檔。如需在託管運算上安裝和執行 代理程式的說明，請參閱 代理程式儲存庫中的讀我檔案。您的啟動命令應至少包含指定 Anywhere 機群 ID AWS 區域和自訂位置和運算名稱的選項。

代理程式會自動為您處理下列任務，因此如果您一直使用指令碼處理這些任務，您可以移除它們：

- 呼叫 `RegisterCompute()` 將運算新增至 Anywhere 機群。
- 當遊戲伺服器連線到 Amazon GameLift Servers 服務時，呼叫 `GetComputeAuthToken()` 來驗證遊戲伺服器。代理程式會管理取得和重新整理身分驗證權杖，以供在運算上執行的所有遊戲伺服器程序使用。
- 根據一組執行時間指示，在運算上啟動新的伺服器程序。
- 為 Anywhere 機群中的運算建立執行期組態。至少，指定遊戲伺服器可執行檔的啟動路徑。您可以使用 Amazon GameLift Servers 主控台或 AWS CLI 來建立或修改機群的執行時間指示。代理程式會執行這些指示，並定期向 Amazon GameLift Servers 服務請求更新。
- 視需要設定或修改遊戲工作階段佇列。建立新的佇列（或更新現有的佇列），並為隨 Amazon GameLift Servers 代理程式部署的 Anywhere 機群指定目的地。
- 測試 代理程式與 Anywhere 機群的整合。檢查 代理程式是否根據執行時間組態正確啟動伺服器程序。

步驟 5：建立受管雲端型機群

建立 Amazon GameLift Servers 受管 EC2 機群以補充您的 Anywhere 機群。如果您在步驟 2 中設定雲端型測試環境以加速開發，請在完成大部分的遊戲開發和測試後，計劃建立受管機群。您需要全受管機群來設定和測試其他設定，例如自動容量擴展。

- 封裝遊戲伺服器建置並上傳至 Amazon GameLift Servers。使用建置檔案、相依性和支援的軟體建立安裝指令碼。您可以搭配 Anywhere 和受管機群使用相同的建置軟體。請參閱 [部署用於 Amazon GameLift Servers 託管的自訂伺服器建置](#)。您可以使用 主控台或 Amazon GameLift Servers CLI 將組建上傳至 AWS。

在上傳建置之前，請先決定 AWS 區域 您要建立受管機群的內容。您必須將組建上傳到相同的區域。如需選擇機群位置的詳細資訊，請參閱 [機群位置](#)。

- 建立受管 EC2 機群。您可以使用 Amazon GameLift Servers 主控台或 AWS CLI 來建立受管機群。當您建立機群時，會 Amazon GameLift Servers 立即開始部署遊戲伺服器組建以進行託管。您可以設定受管機群的許多層面。如需準則，請參閱 [建立 Amazon GameLift Servers 受管 EC2 機群](#)。至少，請執行下列動作：
 - 為機群命名並指定要部署的上傳遊戲組建。

- 選擇機群的隨需執行個體，然後選取機群位置中可用的執行個體類型。Spot 機群是寶貴的選項，但需要額外的設計和組態。
- 使用與 Anywhere 機群搭配使用的類似設定來建立執行期組態。至少，指定遊戲伺服器可執行檔的啟動路徑。
- 指定連接埠設定，以允許傳入流量存取您的遊戲伺服器。
- 將受管機群新增至共用遊戲工作階段佇列。從步驟 4 更新佇列，使其包含受管機群和隨 Amazon GameLift Servers 代理程式部署的 Anywhere 機群的目的地。
- 使用受管機群測試遊戲託管。此時，您應該能夠測試整個託管週期，讓遊戲用戶端請求遊戲工作階段、取得連線資訊，以及成功連線至遊戲工作階段。

步驟 6：自訂您的機群

準備啟動遊戲時，您需要微調託管解決方案。要考慮的一些決策包括：

- 對於 Anywhere 機群，請視需要自動化啟動和關閉運算的程序，包括安裝和執行遊戲伺服器軟體。回收運算有助於確保它們定期更新，而關閉運算可以在不需要時節省成本。
- 如果您的遊戲伺服器需要通訊其他 AWS 資源，請設定 IAM 角色來管理存取。請參閱 [與機群中的其他 AWS 資源通訊](#)。
- 決定要在地理上放置遊戲伺服器的位置。將遠端位置新增至受管機群。請參閱 [自訂您的 Amazon GameLift Servers EC2 受管機群](#)。
- 對於受管機群，請考慮使用 Spot 機群來節省成本。請參閱 [教學課程：使用 Spot 執行個體建立 Amazon GameLift Servers 佇列](#)。
- 透過選取運算資源組態來最佳化機群效能，然後設定執行時間指示，以執行每個運算的最佳伺服器程序數量。同時為 Anywhere 機群和受管機群執行此操作。請參閱 [管理如何 Amazon GameLift Servers 啟動遊戲伺服器](#)。
- 試用受管機群的遊戲工作階段置放選項，包括自訂優先順序設定。請參閱 [自訂遊戲工作階段佇列](#)。
- 對於受管機群，設定自動容量擴展以滿足預期的玩家需求。請參閱 [使用擴展遊戲託管容量 Amazon GameLift Servers](#)。
- 對於 Anywhere 機群，建立機制來處理手動或自動化容量擴展，以滿足預期的玩家需求。
- 視需要設計和實作容錯移轉至其他資源。在其他 AWS 區域中設定待命機群，並視需要修改佇列和自動擴展以處理容錯移轉。
- 設定託管可觀測性工具，包括分析和記錄。請參閱 [監控 Amazon GameLift Servers](#)。建立指標群組來彙總所有託管資源的分析。

- 使用[基礎設施即程式碼 \(IaC\)](#) 自動化部署。請參閱 [使用 管理Amazon GameLift Servers託管資源 AWS CloudFormation](#)。

Amazon GameLift Servers 支援針對任何部署特定的組態使用 AWS CloudFormation 範本。您也可以使用 AWS Cloud Development Kit (AWS CDK) 來定義您的 Amazon GameLift Servers 資源。如需的詳細資訊 AWS CDK，請參閱 [AWS Cloud Development Kit \(AWS CDK\) 開發人員指南](#)。

若要管理 AWS CloudFormation 堆疊的部署，建議您使用持續整合和持續交付 (CI/CD) 工具和服務，例如 AWS CodePipeline。這些工具可協助您在每次建置遊戲伺服器二進位檔時，自動部署 或獲得核准。使用 CI/CD 工具或服務時，新遊戲伺服器版本的資源部署如下所示：

- 建置和測試您的遊戲伺服器二進位檔。
- 將二進位檔案上傳至 Amazon GameLift Servers。
- 使用新建置部署新機群。
- 將新機群新增至您的遊戲工作階段佇列，並移除具有先前建置版本的機群。
- 當具有先前建置的機群不再託管作用中的遊戲工作階段時，請刪除這些機群的 AWS CloudFormation 堆疊。

準備的遊戲 Amazon GameLift Servers

讓您的多玩家遊戲準備好在 上託管Amazon GameLift Servers。將Amazon GameLift Servers託管功能整合至您的遊戲專案，並建置您的遊戲伺服器和用戶端伺服器。設定託管測試環境，以支援快速反覆的遊戲開發和測試。

主題

- [將遊戲與自訂遊戲伺服器整合](#)
- [設計您的遊戲用戶端服務](#)
- [使用 Amazon GameLift ServersAnywhere 設定反覆開發](#)
- [使用 Amazon GameLift Servers Local 測試您的整合](#)
- [新增FlexMatch配對](#)
- [取得Amazon GameLift Servers執行個體的機群資料](#)
- [將遊戲與 整合 Amazon GameLift ServersRealtime](#)

將遊戲與自訂遊戲伺服器整合

Amazon GameLift Servers 提供完整的工具集，可讓您準備要在 上執行的多玩家遊戲和自訂遊戲伺服器Amazon GameLift Servers。Amazon GameLift Servers SDKs包含遊戲用戶端和伺服器與 通訊所需的程式庫Amazon GameLift Servers。如需SDKs以及取得位置的詳細資訊，請參閱 [取得Amazon GameLift Servers開發工具](#)。

本節中的主題包含如何在 上部署 之前，將Amazon GameLift Servers功能新增至遊戲用戶端和遊戲伺服器的詳細指示Amazon GameLift Servers。

主題

- [遊戲用戶端/伺服器與 的互動 Amazon GameLift Servers](#)
- [將遊戲伺服器與 整合 Amazon GameLift Servers](#)
- [將您的遊戲用戶端與 整合Amazon GameLift Servers](#)
- [遊戲引擎和 Amazon GameLift Servers](#)

遊戲用戶端/伺服器與 的互動 Amazon GameLift Servers

Amazon GameLift Servers 託管解決方案中的元件會以特定方式彼此互動，以執行遊戲工作階段來回應玩家的需求。本主題說明當您的遊戲伺服器託管在Amazon GameLift Servers受管 EC2 機群、自我管理Amazon GameLift Servers的 Anywhere 機群或混合解決方案時，元件如何彼此通訊。

託管解決方案元件包括遊戲伺服器、Amazon GameLift Servers服務、用戶端後端服務和遊戲用戶端。遊戲伺服器使用[Amazon GameLift Servers伺服器 SDK](#) 與服務互動Amazon GameLift Servers。後端服務會使用 [Amazon GameLift Servers服務 API](#) (AWS SDK 的一部分) 代表遊戲用戶端與 服務互動。加入遊戲工作階段時，遊戲用戶端會使用遊戲工作階段的唯一 IP 地址和連接埠號碼，直接連線至遊戲工作階段。

主題

- [互動圖表](#)
- [互動行為](#)

互動圖表

下圖說明您的遊戲託管元件如何互動，以便Amazon GameLift Servers服務可以追蹤遊戲伺服器可用性的狀態，並開始遊戲工作階段以回應玩家需求。

互動行為

下列各節說明每個關鍵互動中的事件順序。

初始化遊戲伺服器程序

啟動時，遊戲伺服器程序會建立與服務的通訊，Amazon GameLift Servers並回報其狀態為準備好託管遊戲工作階段。

1. 遊戲伺服器可執行檔的新程序會開始在託管資源上執行。
2. 遊戲伺服器程序會依序呼叫下列伺服器 SDK 操作：
 - a. `InitSDK()` 初始化伺服器 SDK、驗證伺服器程序，以及建立與 Amazon GameLift Servers 服務的通訊。
 - b. `ProcessReady()` 以傳達託管遊戲工作階段的準備程度。此呼叫也會報告程序的連線資訊、遊戲用戶端用來連線至遊戲工作階段的哪些資訊，以及其他資訊。

伺服器程序接著會等待來自 Amazon GameLift Servers 服務的提示。

3. Amazon GameLift Servers 會將伺服器程序的狀態更新為 `ACTIVE` 並可用於託管新的遊戲工作階段。
4. Amazon GameLift Servers 會開始定期呼叫 `onHealthCheck` 函數，向伺服器程序請求運作狀態。這些呼叫會在伺服器程序保持作用中狀態時繼續。伺服器程序必須在一分鐘內回應正常或不良。如果伺服器程序回應狀況不良或沒有回應，Amazon GameLift Servers 服務會在某個時間點變更伺服器程序的作用中狀態，並停止傳送啟動遊戲工作階段的請求。

建立遊戲工作階段

Amazon GameLift Servers 服務會啟動新的遊戲工作階段，以回應玩家的遊戲請求。

1. 使用遊戲用戶端的玩家要求加入遊戲工作階段。根據您的遊戲如何處理玩家聯結程序，遊戲用戶端會將請求傳送至後端服務。
2. 如果玩家加入程序需要啟動新的遊戲工作階段，後端服務會將新遊戲工作階段的請求傳送至 Amazon GameLift Servers 服務。此請求會呼叫服務 API 操作 `StartGameSessionPlacement()`。（或者，後端服務可能會呼叫 `StartMatchmaking()` 或 `CreateGameSession()`。）
3. Amazon GameLift Servers 服務會透過建立狀態為 `PENDING` 的新 `GameSessionPlacement` 票證來回應 `PENDING`。它會將票證資訊傳回後端服務，以便追蹤置放票證狀態，並判斷遊戲工作階段何時準備好供玩家使用。如需詳細資訊，請參閱[設定遊戲工作階段放置的事件通知](#)。
4. Amazon GameLift Servers 服務會啟動遊戲工作階段置放程序。它會識別要查看哪些機群，並搜尋這些機群是否有未託管遊戲工作階段的作用中伺服器程序。在尋找可用的伺服器程序時，Amazon GameLift Servers 服務會執行下列動作：
 - a. 使用遊戲工作階段設定和置放請求中的玩家資料建立 `GameSession` 物件，並將狀態設定為 `ACTIVATING`。
 - b. 提示伺服器程序啟動遊戲工作階段。服務會叫用伺服器程序的 `onStartGameSession` 函數並傳遞 `GameSession` 物件。
 - c. 將伺服器程序的遊戲工作階段數目變更為 1。
5. 伺服器程序會執行其 `onStartGameSession` 函數。當伺服器程序準備好接受玩家連線時，它會呼叫伺服器 SDK 操作 `ActivateGameSession()` 並等待玩家連線。

6. Amazon GameLift Servers 服務會使用伺服器程序的連線資訊（如呼叫中所回報 `ProcessReady()`）更新 `GameSession` 物件，並將遊戲工作階段狀態設定為 `ACTIVE`。它也會將 `GameSessionPlacement` 票證狀態更新為 `FULFILLED`。
7. 後端服務會呼叫 `DescribeGameSessionPlacement()` 來檢查票證狀態並取得遊戲工作階段資訊。當遊戲工作階段處於作用中狀態時，後端服務會通知遊戲用戶端並傳遞遊戲工作階段連線資訊。
8. 遊戲用戶端會使用連線資訊直接連線至遊戲伺服器程序，並加入遊戲工作階段。

將玩家新增至遊戲

遊戲可以選擇性地使用玩家工作階段來追蹤玩家與遊戲工作階段的連線。玩家工作階段可以個別建立，也可以作為遊戲工作階段置放請求的一部分建立。

1. 後端服務會使用遊戲工作階段 ID `CreatePlayerSession()` 呼叫服務 API 操作。
2. Amazon GameLift Servers 服務會檢查遊戲工作階段狀態（必須是 `ACTIVE`），並在遊戲工作階段中尋找開啟的玩家位置。如果插槽可用，則服務會執行下列動作：
 - a. 建立新的 `PlayerSession` 物件，並將狀態設定為 `RESERVED`。
 - b. 使用玩家工作階段資訊回應後端服務請求。
3. 後端服務會將玩家工作階段資訊與遊戲工作階段連線資訊一起傳遞至遊戲用戶端。
4. 遊戲用戶端使用連線資訊和玩家工作階段 ID 直接連線至遊戲伺服器程序，並要求加入遊戲工作階段。
5. 為了回應遊戲用戶端加入嘗試，遊戲伺服器程序會呼叫服務 API 操作 `AcceptPlayerSession()` 來驗證玩家工作階段 ID。伺服器程序接受或拒絕連接。
6. Amazon GameLift Servers 服務會執行下列其中一項操作：
 - a. 如果接受連線，則會將 `PlayerSession` 狀態 Amazon GameLift Servers 設定為 `ACTIVE`，並將傳遞 `PlayerSession` 至遊戲伺服器程序。
 - b. 如果遊戲伺服器程序未在原始 `CreatePlayerSession()` 請求後的特定期間內 `AcceptPlayerSession()` 呼叫玩家工作階段 ID，Amazon GameLift Servers 服務會將 `PlayerSession` 狀態變更為 `TIMEDOUT` 並在遊戲工作階段中重新開啟玩家位置。

移除玩家

對於使用玩家工作階段的遊戲，遊戲伺服器程序會在玩家中斷連線時通知 Amazon GameLift Servers 服務。服務會使用此資訊來追蹤遊戲工作階段中玩家位置的狀態，並允許新玩家使用開放的插槽。

1. 玩家與遊戲工作階段中斷連線。
2. 遊戲伺服器程序會偵測遺失的連線，並呼叫伺服器 SDK 操作 `RemovePlayerSession()`。
3. Amazon GameLift Servers 服務會將玩家工作階段狀態變更為 `COMPLETED`並在遊戲工作階段中重新開啟玩家位置。

關閉遊戲工作階段

在遊戲工作階段結束時，或關閉遊戲工作階段時，伺服器程序會通知 Amazon GameLift Servers 服務遊戲工作階段狀態。

1. 遊戲伺服器程序會結束遊戲工作階段，並透過呼叫伺服器 SDK 操作 `ProcessEnding()` 來啟動程序關閉。
2. Amazon GameLift Servers 服務會執行下列動作：
 - a. 將遊戲工作階段日誌上傳至 Amazon Simple Storage Service (Amazon S3)。
 - b. 將遊戲工作階段狀態變更為 `TERMINATED`。
 - c. 將伺服器程序狀態變更為 `TERMINATED`。
 - d. 根據託管解決方案的設計方式，新可用的託管資源會配置為執行新的遊戲伺服器程序。

將遊戲伺服器與 整合 Amazon GameLift Servers

在 Amazon GameLift Servers 執行個體上部署和執行自訂遊戲伺服器之後，它必須能夠與 Amazon GameLift Servers（以及可能的其他資源）互動。本節說明如何將遊戲伺服器軟體與 整合 Amazon GameLift Servers。

Note

這些指示假設您已建立 `GameLift`，AWS 帳戶 且您擁有現有的遊戲伺服器專案。

本節中的主題說明如何處理下列整合任務：

- 建立 Amazon GameLift Servers 與遊戲伺服器之間的通訊。
- 產生並使用 TLS 憑證，在遊戲用戶端和遊戲伺服器之間建立安全連線。
- 提供遊戲伺服器軟體與其他 AWS 資源互動的許可。
- 允許遊戲伺服器程序取得其執行中機群的相關資訊。

主題

- [Amazon GameLift Servers 新增至您的遊戲伺服器](#)
- [與機群中的其他 AWS 資源通訊](#)

Amazon GameLift Servers 新增至您的遊戲伺服器

本主題說明如何修改遊戲伺服器程式碼，讓遊戲伺服器程序可以與服務通訊Amazon GameLift Servers。對於您計劃部署到Amazon GameLift Servers受管 EC2 機群、受管容器機群或 Anywhere 機群的遊戲伺服器，請使用這些指示。

遊戲伺服器程序會與服務通訊Amazon GameLift Servers，以接收來自服務的指示，並報告伺服器程序運作狀態和遊戲工作階段狀態。如需遊戲託管解決方案元件（遊戲伺服器、後端服務、遊戲用戶端和 Amazon GameLift Servers）之間互動的詳細資訊，請參閱 [遊戲用戶端/伺服器與的互動 Amazon GameLift Servers](#)。

若要準備您的遊戲進行託管，請將適用於的伺服器 SDK Amazon GameLift Servers 新增至您的遊戲伺服器專案。如果您使用的是 Unreal Engine 或 Unity 的Amazon GameLift Servers外掛程式，則伺服器 SDK 已內建並可供使用。伺服器 SDK 提供多種語言。如需遊戲伺服器工具支援的詳細資訊，包括伺服器 SDK，請參閱 [取得Amazon GameLift Servers開發工具](#)。

伺服器開發套件 API 參考：

- [適用於 -- 動作的 C++ 伺服器 SDK Amazon GameLift Servers 5.x](#)
- [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)
- [適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x](#)
- [適用於 Amazon GameLift Servers -- 動作的 Go 伺服器 SDK](#)

初始化伺服器程序

新增程式碼以建立Amazon GameLift Servers與服務的通訊，並在遊戲伺服器程序準備好託管遊戲工作階段時回報。此程式碼必須在任何Amazon GameLift Servers程式碼之前執行。

1. 呼叫來初始化 Amazon GameLift Servers API 用戶端InitSdk()。如果您要準備在Amazon GameLift Servers受管 EC2 機群上執行遊戲伺服器，請使用預設 InitSDK()(C++) (C#) (Unreal) (Go) 且沒有參數。API 用戶端會為您處理 Amazon GameLift Servers服務的連線。

i 如果您要準備要在 Amazon GameLift ServersAnywhere 機群上使用的遊戲伺服器：

InitSdk() 使用下列 呼叫 來初始化 Amazon GameLift Servers API 用戶端ServerParameters：

- 用來連線至遊戲伺服器的 Websocket URL。
- 用於託管遊戲伺服器的程序 ID。
- 託管遊戲伺服器程序的運算 ID。
- 包含 Amazon GameLift ServersAnywhere 運算的機群 ID。
- Amazon GameLift Servers 操作 產生的授權字符合[GetComputeAuthToken](#)。

2. 通知服務遊戲伺服器程序已準備好託管遊戲工作階段。使用下列 呼叫 ProcessReady()([C++](#)) ([C#](#)) ([非真實](#)) ([Go](#)) ProcessParameters。每個遊戲伺服器程序ProcessReady()只能呼叫一次。

- 伺服器程序的連接埠號碼。當伺服器程序啟動遊戲工作階段時，它會提供 Amazon GameLift Servers服務的連接埠，以更新遊戲工作階段資訊。您的遊戲可以擷取此資訊，並將其提供給遊戲用戶端，以用來連線至伺服器程序並加入遊戲工作階段。
- Amazon GameLift Servers 您要為您存放的檔案位置。這些可能包括遊戲工作階段日誌，以及伺服器程序在遊戲工作階段期間產生的其他檔案。雖然 會將這些檔案暫時Amazon GameLift Servers儲存在執行伺服器程序的運算上，但這些檔案只能在執行個體關閉之前使用。您可以透過[Amazon GameLift Servers主控台](#)或呼叫 Amazon GameLift Servers API 操作 [GetGameSessionLogUrl\(\)](#) 來存取儲存的檔案。

i 如果您要準備在Amazon GameLift Servers受管容器機群上使用的遊戲伺服器：

您不需要指定容器機群的日誌參數。反之，請將遊戲工作階段和其他日誌資料傳送至標準輸出。容器機群會自動將所有容器標準輸出擷取為日誌串流。

- 下列回呼函數Amazon GameLift Servers允許 傳送訊息或提示至遊戲伺服器程序。您必須在遊戲伺服器程式碼中實作這些函數。如需詳細資訊，請參閱 ProcessParameters([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 。
- (選用) onHealthCheck – 定期Amazon GameLift Servers呼叫此函數，向伺服器請求運作狀態報告。
- onStartGameSession – Amazon GameLift Servers呼叫此函數以回應用戶端請求 [CreateGameSession\(\)](#)。

- `onProcessTerminate` – Amazon GameLift Servers 強制伺服器程序停止，使其正常關閉。
- (選用) `onUpdateGameSession` – 將更新的遊戲工作階段物件 Amazon GameLift Servers 交付至遊戲伺服器，或提供配對回填請求的狀態更新。[FlexMatch 回填](#) 功能需要此回呼。

您也可以設定遊戲伺服器，使其可以安全地存取您擁有或控制的其他 AWS 資源。如需詳細資訊，請參閱[與機群中的其他 AWS 資源通訊](#)。

(選用) 報告伺服器程序運作狀態

將程式碼新增至遊戲伺服器以實作回呼函數 `onHealthCheck()`。會定期 Amazon GameLift Servers 叫用此回呼方法，以收集運作狀態指標。若要實作此回呼函數，請執行下列動作：

- 評估伺服器程序的運作狀態。例如，如果任何外部相依性失敗，您可以將伺服器程序報告為運作狀態不佳。
- 完成運作狀態評估和在 60 秒內回應回呼。如果在此期間 Amazon GameLift Servers 未收到回應，則會自動將伺服器程序視為運作狀態不佳。
- 傳回布林值：正常為 `true`，不良為 `false`。

如果您未實作運作狀態檢查回呼，則會將伺服器程序 Amazon GameLift Servers 視為正常運作，除非伺服器未回應。

Amazon GameLift Servers 服務使用伺服器程序運作狀態來結束運作狀態不佳的程序，並清除資源。如果伺服器程序持續回報為運作狀態不佳，或連續三次未回應運作狀態檢查，則服務可能會關閉程序並啟動新的程序。服務會收集機群伺服器程序運作狀態的指標。

(選用) 取得 TLS 憑證

如果伺服器程序是在已啟用 TLS 憑證產生功能的機群上執行，則您可以擷取 TLS 憑證來與遊戲用戶端建立安全連線，並加密用戶端伺服器通訊。憑證的副本會存放在執行個體上。若要取得檔案位置，請呼叫 `GetComputeCertificate()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#))。

啟動遊戲工作階段

新增程式碼以實作回呼函數 `onStartGameSession`。會 Amazon GameLift Servers 叫用此回呼，以在伺服器程序上啟動遊戲工作階段。

`onStartGameSession` 函數會將 [GameSession](#) 物件作為輸入參數。此物件包含關鍵遊戲工作階段資訊，例如玩家數量上限。它也可以包含遊戲資料和玩家資料。函數實作應執行下列任務：

- 根據GameSession屬性啟動動作以建立新的遊戲工作階段。遊戲伺服器至少必須關聯遊戲工作階段 ID，遊戲用戶端會在連線至伺服器程序時參考此 ID。
- 視需要處理遊戲資料和玩家資料。此資料位於 GameSession 物件中。
- 當新的遊戲工作階段準備好接受玩家時，請通知 Amazon GameLift Servers 服務。呼叫伺服器 API 操作 `ActivateGameSession()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#))。為了回應成功的呼叫，服務會將遊戲工作階段狀態變更為 ACTIVE。

(選用) 驗證新的玩家

如果您要追蹤玩家工作階段的狀態，請新增程式碼，以在新玩家連線到遊戲伺服器時驗證新玩家。會 Amazon GameLift Servers 追蹤目前玩家和可用的遊戲工作階段位置。

為了進行驗證，嘗試加入遊戲工作階段的遊戲用戶端必須包含玩家工作階段 ID。當您的遊戲透過呼叫 [StartGameSessionPlacement\(\)](#) 或 [StartMatchmaking\(\)](#) 來啟動新的遊戲工作階段時，Amazon GameLift Servers 會產生此 ID。在這些請求上，遊戲工作階段中的開放位置會保留給玩家工作階段。

當遊戲伺服器程序收到遊戲用戶端連線請求時，它會使用玩家工作階段 ID 呼叫 `AcceptPlayerSession()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#))。為了回應，Amazon GameLift Servers 會驗證玩家工作階段 ID 是否對應到遊戲工作階段中預留的開放位置。在 Amazon GameLift Servers 驗證玩家工作階段 ID 之後，伺服器程序會接受連線。然後，玩家可以加入遊戲工作階段。如果 Amazon GameLift Servers 未驗證玩家工作階段 ID，則伺服器程序會拒絕連線。

(選用) 報告玩家工作階段結束

如果您要追蹤玩家工作階段的狀態，請新增程式碼，以在玩家離開遊戲工作階段 Amazon GameLift Servers 時通知。此程式碼應在伺服器程序偵測到中斷連線時執行。Amazon GameLift Servers 會使用此通知來追蹤遊戲工作階段中的目前玩家和可用位置。

若要處理程式碼中捨棄的連線，請使用對應的玩家工作階段 ID，將呼叫新增至伺服器 API 操作 `RemovePlayerSession()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#))。

結束遊戲工作階段

將程式碼新增至伺服器程序關閉序列，以在 Amazon GameLift Servers 遊戲工作階段結束時通知。若要回收和重新整理託管資源，請在遊戲工作階段完成後關閉每個伺服器程序。

在伺服器程序關閉代碼開始時，呼叫伺服器 API 操作 `ProcessEnding()` ([C++](#)) ([C#](#)) ([非真實](#)) ([Go](#))。此呼叫 Amazon GameLift Servers 會通知伺服器程序正在關閉。Amazon GameLift Servers 會將遊戲工作階段狀態和伺服器程序狀態變更為 TERMINATED。呼叫後 `ProcessEnding()`，程序可以安全地關閉。

回應伺服器程序關閉通知

新增程式碼以關閉伺服器程序，以回應來自 Amazon GameLift Servers 服務的通知。當伺服器程序持續回報運作狀態不佳，或伺服器程序正在終止的執行個體時，服務會傳送此通知。Amazon GameLift Servers 可以在容量縮減事件中停止執行個體，或回應 Spot 執行個體中斷。Spot 執行個體中斷提供兩分鐘的通知，讓伺服器有時間正常中斷與玩家的連線、保留遊戲狀態資料，以及執行其他清除任務。

若要處理關閉通知，請對遊戲伺服器程式碼進行下列變更：

- 實作回呼函數 `onProcessTerminate()` ([C++](#)) ([C#](#)) ([非真實](#)) ([Go](#))。此函數應呼叫關閉伺服器程序的程式碼。
- 從遊戲伺服器關閉碼呼叫伺服器 API 操作 `GetTerminationTime()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#))。如果 Amazon GameLift Servers 已發出停止伺服器程序的呼叫，則 `GetTerminationTime()` 傳回預估的終止時間。
- 在遊戲伺服器關閉程式碼開始時，呼叫伺服器 API 操作 `ProcessEnding()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#))。此呼叫會通知 Amazon GameLift Servers 服務該伺服器程序正在關閉。然後，服務會將伺服器程序狀態變更為 `TERMINATED`。呼叫後 `ProcessEnding()`，程序可以安全地關閉。

與機群中的其他 AWS 資源通訊

當您建立遊戲伺服器組建以在機 Amazon GameLift Servers 群上部署時，您可能會希望遊戲組建中的應用程式直接且安全地與您擁有的其他 AWS 資源通訊。由於 Amazon GameLift Servers 會管理您的遊戲託管機群，因此您必須提供這些資源與服務的 Amazon GameLift Servers 有限存取權。

一些範例案例包括：

- 使用 Amazon CloudWatch 代理程式從受管 EC2 機群和 Anywhere 機群收集指標、日誌和追蹤。
- 將執行個體日誌資料傳送至 Amazon CloudWatch Logs。
- 取得存放在 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的遊戲檔案。
- 讀取和寫入存放在 Amazon DynamoDB 資料庫或其他資料儲存服務的遊戲資料（例如遊戲模式或庫存）。
- 使用 Amazon Simple Queue Service (Amazon SQS) 將訊號直接傳送至執行個體。
- 存取在 Amazon Elastic Compute Cloud (Amazon EC2) 上部署和執行的自訂資源。

Amazon GameLift Servers 支援這些建立存取的方法：

- [使用 IAM 角色存取 AWS 資源](#)

- [使用 VPC 對等互連存取 AWS 資源](#)

使用 IAM 角色存取 AWS 資源

使用 IAM 角色指定誰可以存取您的 資源，並設定該存取的限制。受信任方可以「擔任」角色並取得臨時安全登入資料，以授權他們與資源互動。當各方提出與資源相關的 API 請求時，他們必須包含登入資料。

若要設定由 IAM 角色控制的存取，請執行下列任務：

1. [建立 IAM 角色](#)
2. [修改應用程式以取得登入資料](#)
3. [將機群與 IAM 角色建立關聯](#)

建立 IAM 角色

在此步驟中，您會建立具有一組許可的 IAM 角色，以控制對 AWS 資源的存取，以及授予使用角色許可 Amazon GameLift Servers 權限的信任政策。

如需如何設定 IAM 角色的指示，請參閱 [設定的 IAM 服務角色 Amazon GameLift Servers](#)。建立許可政策時，請選擇應用程式需要使用的特定服務、資源和動作。最佳實務是盡可能限制許可的範圍。

建立角色之後，請記下角色的 Amazon Resource Name (ARN)。您需要在機群建立期間的角色 ARN。

修改應用程式以取得登入資料

在此步驟中，您將應用程式設定為取得 IAM 角色的安全登入資料，並在與 AWS 資源互動時使用它們。請參閱下表，以根據 (1) 應用程式類型，以及 (2) 遊戲用來與 通訊的伺服器 SDK 版本，決定如何修改您的應用程式 Amazon GameLift Servers。

	遊戲伺服器應用程式	其他應用程式
使用伺服器 SDK 5.x 版	<code>GetFleetRoleCredentials()</code> 從遊戲伺服器程式碼呼叫伺服器 SDK 方法。	將程式碼新增至應用程式，從機群執行個體上的共用檔案提取登入資料。
使用伺服器 SDK 第 4 版或更早期版本	AssumeRole 使用角色 ARN 呼叫 AWS Security Token Service (AWS STS)。	AssumeRole 使用角色 ARN 呼叫 AWS Security Token Service (AWS STS)。

對於與伺服器 SDK 5.x 整合的遊戲，此圖說明部署的遊戲組建中的應用程式如何取得 IAM 角色的登入資料。

呼叫 **GetFleetRoleCredentials()** (伺服器 SDK 5.x)

在應該已與Amazon GameLift Servers伺服器 SDK 5.x 整合的遊戲伺服器程式碼中，呼叫 **GetFleetRoleCredentials**([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 以擷取一組臨時登入資料。當登入資料過期時，您可以使用對的另一個呼叫來重新整理它們**GetFleetRoleCredentials**。

使用共用登入資料 (伺服器 SDK 5.x)

對於使用伺服器 SDK 5.x 搭配遊戲伺服器組建部署的非伺服器應用程式，請新增程式碼以取得和使用存放在共用檔案中的登入資料。會為每個機群執行個體Amazon GameLift Servers產生登入資料設定檔。登入資料可供執行個體上的所有應用程式使用。Amazon GameLift Servers會持續重新整理臨時登入資料。

您必須設定機群，以在機群建立時產生共用的登入資料檔案。

在需要使用共用登入資料檔案的每個應用程式中，指定檔案位置和設定檔名稱，如下所示：

Windows：

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "C:\\Credentials\\credentials"
```

Linux：

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "/local/credentials/credentials"
```

範例：設定 CloudWatch 代理程式以收集Amazon GameLift Servers機群執行個體的指標

如果您想要使用 Amazon CloudWatch 代理程式從機Amazon GameLift Servers群收集指標、日誌和追蹤，請使用此方法授權代理程式將資料傳送到您的帳戶。在此案例中，請採取下列步驟：

1. 擷取或寫入 CloudWatch 代理程式config.json檔案。
2. 更新代理程式common-config.toml的檔案，以識別登入資料檔案名稱和設定檔名稱，如上所述。

3. 設定遊戲伺服器建置安裝指令碼來安裝和啟動 CloudWatch 代理程式。

使用 **AssumeRole()** (伺服器 SDK 4)

將程式碼新增至您的應用程式以擔任 IAM 角色，並取得與您的 AWS 資源互動的登入資料。在具有伺服器 SDK 4 或更早版本的 Amazon GameLift Servers 機群執行個體上執行的任何應用程式都可以擔任 IAM 角色。

在應用程式程式碼中，在存取 AWS 資源之前，應用程式必須呼叫 AWS Security Token Service (AWS STS) [AssumeRole](#) API 操作並指定角色 ARN。此操作會傳回一組臨時登入資料，授權應用程式存取 AWS 資源。如需詳細資訊，請參閱《IAM 使用者指南》中的[將臨時登入資料與 AWS 資源搭配使用](#)。

將機群與 IAM 角色建立關聯

在您建立 IAM 角色並更新遊戲伺服器建置中的應用程式以取得和使用存取憑證之後，您可以部署機群。當您設定新機群時，請設定下列參數：

- [InstanceRoleArn](#) – 將此參數設定為 IAM 角色的 ARN。
- [InstanceRoleCredentialsProvider](#) – 若要提示 為每個機群執行個體 Amazon GameLift Servers 產生共用登入資料檔案，請將此參數設定為 SHARED_CREDENTIAL_FILE。

建立機群時，您必須設定這些值。它們稍後無法更新。

使用 VPC 對等互連存取 AWS 資源

您可以使用 Amazon Virtual Private Cloud (Amazon VPC) 對等互連，在 Amazon GameLift Servers 執行個體上執行的應用程式與其他 AWS 資源之間進行通訊。VPC 是您定義的虛擬私有網路，其中包含透過管理的一組資源 AWS 帳戶。每個 Amazon GameLift Servers 叢集都有自己的 VPC。使用 VPC 對等互連，您可以在機群的 VPC 與其他 AWS 資源之間建立直接網路連線。

Amazon GameLift Servers 簡化為遊戲伺服器設定 VPC 對等互連的程序。它會處理對等請求、更新路由表，並視需要設定連線。如需如何為遊戲伺服器設定 VPC 對等互連的說明，請參閱[適用於 Amazon GameLift Servers 的 VPC 對等互連](#)。

將您的遊戲用戶端與 整合 Amazon GameLift Servers

本節中的主題說明您可以新增至後端服務的受管 Amazon GameLift Servers 功能。後端服務會處理下列任務：

- 從 請求有關作用中遊戲工作階段的資訊 Amazon GameLift Servers。

- 將玩家加入現有的遊戲工作階段。
- 建立新的遊戲工作階段，並將玩家加入其中。
- 變更現有遊戲工作階段的中繼資料。

如需遊戲用戶端如何與 互動Amazon GameLift Servers以及在 上執行的遊戲伺服器的詳細資訊Amazon GameLift Servers，請參閱 [遊戲用戶端/伺服器與 的互動 Amazon GameLift Servers](#)。

先決條件

- AWS 帳戶。
- 上傳至 的遊戲伺服器組建Amazon GameLift Servers。
- 託管遊戲的機群。

主題

- [Amazon GameLift Servers 新增至您的遊戲用戶端](#)
- [產生玩家 IDs](#)

Amazon GameLift Servers 新增至您的遊戲用戶端

整合Amazon GameLift Servers到需要遊戲工作階段資訊的遊戲元件、建立新的遊戲工作階段，以及將玩家新增至遊戲。根據您的遊戲架構，此功能位於後端服務中，可處理玩家身分驗證、配對或遊戲工作階段放置等任務。

Note

如需如何為您的遊戲設定配對的詳細資訊，請參閱 [Amazon GameLift ServersFlexMatch開發人員指南](#)。

在後端服務Amazon GameLift Servers上設定

新增程式碼以初始化Amazon GameLift Servers用戶端和存放金鑰設定。此程式碼必須在依賴 的任何程式碼之前執行Amazon GameLift Servers。

1. 設定用戶端組態。使用預設用戶端組態或建立自訂用戶端組態物件。如需詳細資訊，請參閱 [AWS::Client::ClientConfiguration](#)(C++) 或 [AmazonGameLiftConfig](#) (C#)。

用戶端組態會指定聯絡時要使用的目標區域和端點 Amazon GameLift Servers。區域識別要使用的一組已部署資源（機群、佇列和配對建構器）。預設用戶端組態會將位置設定為美國東部（維吉尼亞北部）區域。若要使用任何其他區域，請建立自訂組態。

2. 初始化 Amazon GameLift Servers 用戶端。使用 [Aws::GameLift::GameLiftClient\(\)](#) (C++) 或 [AmazonGameLiftClient\(\)](#) (C#) 搭配預設用戶端組態或自訂用戶端組態。
3. 新增機制，為每個玩家產生唯一的識別符。如需詳細資訊，請參閱[產生玩家 IDs](#)。
4. 收集並存放下列資訊：
 - 目標機群 – 許多 Amazon GameLift Servers API 請求必須指定機群。若要這樣做，請使用指向目標機群的機群 ID 或別名 ID。最佳實務是使用機群別名，讓您可以將玩家從一個機群切換到另一個機群，而無需更新您的後端服務。
 - 目標佇列 – 對於使用多機群佇列放置新遊戲工作階段的遊戲，請指定要使用的佇列名稱。
 - AWS 登入資料 – 對的所有呼叫 Amazon GameLift Servers 都必須為託管遊戲的提供登入 AWS 帳戶資料。您可以透過建立玩家使用者來取得這些登入資料，如中所述[為您的遊戲設定程式設計存取](#)。根據您管理玩家使用者存取權的方式，請執行下列動作：
 - 如果您使用角色來管理玩家使用者許可，請在呼叫 Amazon GameLift Servers API 之前新增程式碼以擔任角色。擔任角色的請求會傳回一組臨時安全登入資料。如需詳細資訊，請參閱《[IAM 使用者指南](#)》中的[切換到 IAM 角色 \(AWS API\)](#)。
 - 如果您有長期安全登入資料，請將您的程式碼設定為尋找和使用儲存的登入資料。請參閱《[AWS SDKs 和工具參考指南](#)》中的[使用中的長期憑證進行身分驗證](#)。如需儲存登入資料的資訊，請參閱 [\(C++\)](#) 和 [\(.NET\)](#) 的 AWS API 參考。
 - 如果您有臨時安全登入資料，請新增程式碼，以使用 AWS Security Token Service (AWS STS) 定期重新整理登入資料，如 IAM 使用者指南中的[使用臨時安全登入資料搭配 AWS SDKs](#) 所述。程式碼必須在舊登入資料過期之前請求新的登入資料。

取得遊戲工作階段

新增程式碼以探索可用的遊戲工作階段，並管理遊戲工作階段設定和中繼資料。

搜尋作用中的遊戲工作階段

使用 [SearchGameSessions](#) 取得特定遊戲工作階段、所有作用中工作階段或符合一組搜尋條件的工作階段的相關資訊。此呼叫為每個符合搜尋請求的活動遊戲工作階段傳回一個 [GameSession](#) 物件。

使用搜尋條件取得篩選過的列表，列出可供玩家加入的活動遊戲工作階段。例如，您可以篩選工作階段，如下所示：

- 排除已滿的遊戲工作階段：CurrentPlayerSessionCount = MaximumPlayerSessionCount。
- 根據工作階段執行的時間長度選擇遊戲工作階段：評估 CreationTime。
- 根據自訂遊戲屬性尋找遊戲工作階段：gameSessionProperties.gameMode = "brawl"。

管理遊戲工作階段

使用以下任意一項操作來擷取或更新遊戲工作階段資訊。

- [DescribeGameSessionDetails\(\)](#) – 取得遊戲工作階段的保護狀態，以及遊戲工作階段資訊。
- [UpdateGameSession\(\)](#) – 視需要變更遊戲工作階段的中繼資料和設定。
- [GetGameSessionLogUrl](#) – 存取儲存的遊戲工作階段日誌。

建立遊戲工作階段

加入用於在已部署的機群中啟動新遊戲工作階段並使其可供玩家加入的程式碼。建立遊戲工作階段有兩種選項，取決於您要在多個 AWS 區域 或單一區域中部署遊戲。

在多個位置佇列中建立遊戲工作階段

使用 [StartGameSessionPlacement](#) 在佇列中放置新遊戲工作階段請求。若要使用此操作，請建立佇列。這會決定 Amazon GameLift Servers 放置新遊戲工作階段的位置。如需佇列及其使用方式的詳細資訊，請參閱 [使用 Amazon GameLift Servers 佇列管理遊戲工作階段置放](#)。

建立遊戲工作階段置放時，請指定要使用的佇列名稱、遊戲工作階段名稱、並行玩家數量上限，以及選用的一組遊戲屬性。您也可以選擇性地提供玩家清單，以自動加入遊戲工作階段。如果您包含相關區域的玩家延遲資料，則 Amazon GameLift Servers 會使用此資訊，將新的遊戲工作階段放置在為玩家提供理想遊戲體驗的機群上。

若要取得準確的延遲測量，請使用 Amazon GameLift Servers 的 UDP ping 信標。這些端點可讓您測量玩家裝置與潛在託管位置之間的實際 UDP 網路延遲，從而做出比使用 ICMP ping 更準確的置放決策。如需使用 UDP ping 信標測量延遲的詳細資訊，請參閱 [UDP ping 信標](#)。

遊戲工作階段置放為非同步程序。提出請求後，您可以讓請求成功或逾時。您也可以隨時使用 [StopGameSessionPlacement](#) 來取消請求。若要檢查置放請求的狀態，請呼叫 [DescribeGameSessionPlacement](#)。

在特定機群上建立遊戲工作階段

使用 [CreateGameSession](#) 在特定機群中建立新工作階段。這一個同步操作的成功與否取決於該機群是否擁有託管新遊戲工作階段所需的資源。在 Amazon GameLift Servers 建立新的遊戲工作階段並傳回 [GameSession](#) 物件後，您可以將玩家加入其中。

當您使用此操作時，請提供機群 ID 或別名 ID、工作階段名稱，以及遊戲的並行玩家數量上限。您可以選擇包括一組遊戲屬性。遊戲屬性是在金鑰/值對的陣列中定義。

如果您使用 Amazon GameLift Servers 資源保護功能來限制一個玩家可以建立的遊戲工作階段數量，請提供遊戲工作階段建立者的玩家 ID。

將玩家加入遊戲工作階段

新增程式碼以保留作用中遊戲工作階段中的玩家位置，並將遊戲用戶端連接到遊戲工作階段。

1. 在遊戲工作階段中預留玩家位置

若要預留玩家位置，請在遊戲工作階段中新建一個玩家工作階段。如需玩家工作階段的詳細資訊，請參閱 [玩家如何連接到遊戲](#)。

有兩種方式可以建立新的玩家工作階段：

- 使用 [StartGameSessionPlacement](#) 為新遊戲工作階段中的一個或多個玩家預留位置。
- 透過遊戲工作階段 ID 使用 [CreatePlayerSession](#) 或 [CreatePlayerSessions](#) 為一個或多個玩家預留遊戲位置。

Amazon GameLift Servers 首先驗證遊戲工作階段是否接受新的玩家，並且有可用的玩家位置。如果成功，會為玩家 Amazon GameLift Servers 保留位置、建立新的玩家工作階段，並傳回 [PlayerSession](#) 物件。此物件包含遊戲用戶端連線到遊戲工作階段所需的 DNS 名稱、IP 地址和連接埠。

玩家工作階段請求必須包括每個玩家的唯一 ID。如需詳細資訊，請參閱 [產生玩家 IDs](#)。

玩家工作階段可以包含一組自訂玩家資料。此資料存放在新建立的玩家工作階段物件中，您可以透過呼叫 [DescribePlayerSessions\(\)](#) 來擷取。當玩家直接連線至遊戲工作階段時，Amazon GameLift Servers 也會將此物件傳遞至遊戲伺服器。請求多個玩家工作階段時，請為請求中映射到玩家 ID 的每個玩家提供玩家資料字串。

2. 連線至遊戲工作階段

將程式碼加入到遊戲用戶端來擷取包含遊戲工作階段之連線資訊的 `PlayerSession` 物件。使用此資訊建立與伺服器的直接連線。

- 您可以使用指定的連接埠和指派給伺服器程序的 DNS 名稱或 IP 地址來連線。
- 如果您的機群已啟用 TLS 憑證產生，請使用 DNS 名稱和連接埠進行連線。
- 如果您的遊戲伺服器驗證傳入玩家連線，則參考玩家工作階段 ID。

進行連線後，遊戲用戶端和伺服器程序會直接通訊，而不需要 Amazon GameLift Servers。伺服器會維持與的通訊，Amazon GameLift Servers以報告玩家連線狀態、運作狀態等。如果遊戲伺服器驗證傳入的玩家，則會驗證玩家工作階段 ID 是否符合遊戲工作階段中的預留位置，並接受或拒絕玩家連線。當玩家中斷連線時，伺服器程序會報告中斷的連線。

使用遊戲工作階段屬性

您的遊戲用戶端可以使用遊戲屬性將資料傳遞至遊戲工作階段。遊戲屬性是遊戲伺服器可以新增、讀取、列出和變更的鍵值組。您可以在建立新的遊戲工作階段時傳入遊戲屬性，或稍後在遊戲工作階段作用中時傳入遊戲屬性。遊戲工作階段最多可包含 16 個遊戲屬性。您無法刪除遊戲屬性。

例如，您的遊戲提供以下難度等級：Novice、Intermediate、Easy和 Expert。玩家選擇 Easy，然後開始遊戲。您的遊戲用戶端Amazon GameLift Servers會使用 `StartGameSessionPlacement` 或 向 請求新的遊戲工作階段，`CreateGameSession`如前幾節所述。在請求中，用戶端會傳遞此：`{"Key": "Difficulty", "Value": "Easy"}`。

為了回應請求，Amazon GameLift Servers會建立包含指定遊戲屬性的GameSession物件。Amazon GameLift Servers然後，會指示可用的遊戲伺服器啟動新的遊戲工作階段並傳遞GameSession物件。遊戲伺服器使用 Difficulty的 啟動遊戲工作階段Easy。

進一步了解

- [GameProperty 資料類型](#)
- [SearchGameSessions\(\) 範例](#)
- [UpdateGameSession\(\) GameProperties 參數](#)

產生玩家 IDs

Amazon GameLift Servers 使用玩家工作階段來代表連接到遊戲工作階段的玩家。Amazon GameLift Servers會在每次玩家使用與 整合的遊戲用戶端連接到遊戲工作階段時建立玩家工作階段Amazon GameLift Servers。當玩家離開遊戲時，玩家工作階段會結束。Amazon GameLift Servers不會重複使用玩家工作階段。

下列程式碼範例會隨機產生唯一的玩家 IDs：

```
bool includeBrackets = false;
bool includeDashes = true;
string playerId = AZ::Uuid::CreateRandom().ToString<string>(includeBrackets,
    includeDashes);
```

如需玩家工作階段的詳細資訊，請參閱 [Amazon GameLift Servers 主控台中的遊戲和玩家工作階段](#)。

遊戲引擎和 Amazon GameLift Servers

您可以使用受管 Amazon GameLift Servers 服務搭配大多數支援 C++ 或 C# 程式庫的主要遊戲引擎，包括 O3DE、Unreal Engine 和 Unity。建立您遊戲需要的版本；請參閱建立說明和最低需求的各版本 README 檔案。如需可用的 Amazon GameLift Servers 開發套件、支援的開發平台與作業系統等更多資訊，請參閱 [取得 Amazon GameLift Servers 開發工具](#) 的遊戲伺服器內容。

除本主題中提供的特定引擎資訊外，請參考下列主題以獲得有關在您的遊戲伺服器、用戶端和服務中整合 Amazon GameLift Servers 的更多說明：

- [Amazon GameLift Servers 新增至您的遊戲伺服器](#) – 在遊戲伺服器中整合 Amazon GameLift Servers 的詳細說明。
- [Amazon GameLift Servers 新增至您的遊戲用戶端](#) – 整合到遊戲用戶端或服務中的詳細說明，包括建立遊戲工作階段和將玩家加入遊戲。

O3DE

遊戲伺服器

Amazon GameLift Servers 使用 [Amazon GameLift Servers 適用於 C++ 的伺服器 SDK](#)，準備您的遊戲伺服器在上託管。請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)，以取得將所需功能整合到遊戲伺服器的幫助。

遊戲用戶端與服務

啟用遊戲用戶端和/或遊戲服務與 Amazon GameLift Servers 服務進行互動，如查詢可用遊戲工作階段或建立新工作階段，以及將玩家加入遊戲。適用於 [AWS C++ 的 SDK](#) 提供核心用戶端功能。若要 Amazon GameLift Servers 整合到您的 O3DE 遊戲專案，請參閱 [Amazon GameLift Servers 新增至 O3DE 遊戲用戶端和伺服器](#) 和 [Amazon GameLift Servers 新增至您的遊戲用戶端](#)。

虛擬引擎

遊戲伺服器

將[Amazon GameLift Servers適用於 Unreal Engine 的伺服器 SDK](#) Amazon GameLift Servers新增至您的專案，並實作必要的伺服器功能，以準備在 上託管的遊戲伺服器。有關設定 Unreal Engine 外掛程式和加入 Amazon GameLift Servers 程式碼的協助，請參閱 [Amazon GameLift Servers 整合至 Unreal Engine 專案](#)。

遊戲用戶端與服務

啟用遊戲用戶端和/或遊戲服務與 Amazon GameLift Servers 服務進行互動，如查詢可用遊戲工作階段或建立新工作階段，以及將玩家加入遊戲。適用於 [AWS C++ 的 SDK](#) 提供核心用戶端功能。若要將 Amazon GameLift Servers 整合到 Unreal Engine 遊戲專案中，請參閱 [Amazon GameLift Servers 新增至您的遊戲用戶端](#)。

Unity

遊戲伺服器

將[Amazon GameLift Servers適用於 C# 的伺服器 SDK](#) Amazon GameLift Servers新增至您的專案，並實作必要的伺服器功能，以準備在 上託管的遊戲伺服器。有關設定 Unity 和新增 Amazon GameLift Servers 程式碼的協助，請參閱 [Amazon GameLift Servers 整合至 Unity 專案](#)。

遊戲用戶端與服務

啟用遊戲用戶端和/或遊戲服務與 Amazon GameLift Servers 服務進行互動，如查詢可用遊戲工作階段或建立新工作階段，以及將玩家加入遊戲。核心用戶端功能在 [適用於 .NET 的 AWS SDK](#) 中提供。若要將 Amazon GameLift Servers 整合到 Unity 遊戲專案中，請參閱 [Amazon GameLift Servers 新增至您的遊戲用戶端](#)。

其他引擎

有關遊戲伺服器和用戶端可使用的 Amazon GameLift Servers 開發套件的完整列表，請參閱 [the section called “取得開發工具”](#)。

Amazon GameLift Servers 新增至 O3DE 遊戲用戶端和伺服器

您可以使用 O3DE，這是一種開放原始碼、跨平台的即時 3D 引擎，來建立高效能互動式體驗，包括遊戲和模擬。O3DE 轉譯器和工具包裝在模組化架構中，您可以使用您偏好的開發工具來修改和擴展。

模組化架構使用 Gem，其中包含具有標準界面和資產的程式庫。選取您自己的 Gem 套件，根據您的需求選擇要新增的功能。

Amazon GameLift Servers Gem 套件提供下列功能：

Amazon GameLift Servers 整合

擴展 O3DE 網路層並讓多玩家 Gem 套件與Amazon GameLift Servers專用伺服器解決方案搭配使用的架構。Gem 套件提供與適用於的 [伺服器 SDK Amazon GameLift Servers](#) 和 SDK AWS 用戶端（以呼叫Amazon GameLift Servers服務本身）的整合。

建置和套件管理

封裝和選擇性上傳專用伺服器建置和 AWS Cloud Development Kit (AWS CDK) (AWS CDK) 應用程式以設定和更新資源的指示。

Amazon GameLift Servers Gem 套件設定

請遵循本節中的程序，在 O3DE 中設定 Amazon GameLift Servers Gem。

先決條件

- 為設定 AWS 您的帳戶Amazon GameLift Servers。如需詳細資訊，請參閱[設定 AWS 帳戶](#)。
- 設定 O3DE 的 AWS 登入資料。如需詳細資訊，請參閱[設定 AWS 登入資料](#)。
- 設定 AWS CLI 和 AWS CDK。如需詳細資訊，請參閱 [AWS Command Line Interface](#)和 [AWS Cloud Development Kit \(AWS CDK\)](#)。

開啟 Amazon GameLift Servers Gem 套件及其相依性

1. 開啟專案管理員。
2. 開啟專案下方的選單，然後選擇編輯專案設定...
3. 選擇設定 Gem。
4. 開啟 Amazon GameLift Servers Gem 套件和下列相依 Gem 套件：
 - [AWS 核心 Gem](#) – 提供架構以在 O3DE AWS 服務中使用。
 - [多玩家 Gem](#) 套件 – 透過擴展聯網架構提供多玩家功能。

包含 Amazon GameLift Servers Gem 靜態程式庫

1. 將 `Gem::AWSGameLift.Server.Static` 作為 `BUILD_DEPENDENCIES` 納入專案伺服器目標。

```
ly_add_target(
  NAME YourProject.Server.Static STATIC
  ...
  BUILD_DEPENDENCIES
    PUBLIC
    ...
    PRIVATE
    ...
    Gem::AWSGameLift.Server.Static
)
```

2. 將 `AWSGameLiftService` 設定為專案伺服器系統元件所需的。

```
void
YourProjectServerSystemComponent::GetRequiredServices(AZ::ComponentDescriptor::DependencyArray&
required)
{
  ...
  required.push_back(AZ_CRC_CE("AWSGameLiftServerService"));
  ...
}
```

3. (選用) 若要以 C++ 提出 Amazon GameLift Servers 服務請求，請在用戶端目標 `BUILD_DEPENDENCIES` 的 `Gem::AWSGameLift.Client.Static` 中包含。

```
ly_add_target(
  NAME YourProject.Client.Static STATIC
  ...
  BUILD_DEPENDENCIES
    PUBLIC
    ...
    PRIVATE
    ...
    Gem::AWSCore.Static
    Gem::AWSGameLift.Client.Static
}
```


整合您的遊戲和專用伺服器

使用工作階段管理[整合，在遊戲和專用遊戲伺服器中管理遊戲工作階段](#)。若要支援 FlexMatch，請參閱[FlexMatch 整合](#)。

Amazon GameLift Servers 整合至 Unreal Engine 專案

了解如何將適用於 Unreal Engine 的 Amazon GameLift Servers SDK 整合到您的遊戲專案中，以存取完整的伺服器 SDK 功能集。

秘訣

若要加快部署速度，請嘗試 Unreal Engine Amazon GameLift Servers 的獨立外掛程式。它提供引導式 UI 工作流程，以最少的設定快速部署遊戲伺服器，因此您可以嘗試執行中的遊戲元件。請參閱 [Amazon GameLift Servers Unreal Engine 的外掛程式](#)。

其他資源：

- [適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x](#)
- [取得 Amazon GameLift Servers 開發工具](#)

安裝適用於 Unreal 的伺服器 SDK

從 [GitHub](#) 取得適用於 Unreal Engine 的開放原始碼 Amazon GameLift Servers SDK。儲存庫的讀我檔案包含先決條件和安裝指示。

設定建置目標和模組規則

修改您的遊戲專案檔案，以正確產生建置元件，以搭配使用 Amazon GameLift Servers。

若要新增用戶端和伺服器建置目標：

1. 開啟遊戲專案的程式碼檔案，並找到 `.../Games/[your application name]Source/[your application name]Target.cs` 檔案。範例：`.../Source/GameLiftUnrealAppTarget.cs`。（如果您使用 Visual Studio，請開啟專案.sln的檔案。）
2. 複製此檔案以在 Source/目錄中建立新的兩個目標檔案。
 - 用戶端目標 – 將新檔案重新命名為 `[your application name]Client.Target.cs`。編輯內容以更新類別名稱和目標類型值，如下列範例程式碼所示：

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

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

- 伺服器目標 – 將新檔案重新命名為 *[your application name]*Server.Target.cs。編輯內容以更新類別名稱和目標類型值，如下列範例程式碼所示：

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

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

3. 重新產生專案檔案。如果您使用的是 Visual Studio，您可以在遊戲專案的 .uproject 檔案上按一下滑鼠右鍵，然後選取產生 Visual Studio 專案檔案。

若要更新遊戲專案模組規則：

更新遊戲專案的模組規則，以依賴外掛程式。

1. 開啟遊戲專案的程式碼檔案，並找到 `.../Games/[your application name]Source/[your application name].Build.cs` 檔案。範例：`.../Source/GameLiftUnrealApp.Build.cs`。（如果您使用 Visual Studio，請開啟專案 `.sln` 的檔案。）
2. 找到 `ModuleRules` 類別並更新，如下列範例程式碼所示：

```
using UnrealBuildTool;

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

3. 建立新的目標檔案並修改模組規則後，請重建您的遊戲專案。

將遊戲託管功能新增至您的伺服器程式碼

安裝和設定伺服器 SDK 後，下一步是將遊戲託管功能整合到您的伺服器程式碼。伺服器 SDK 可讓您的遊戲伺服器與服務通訊 Amazon GameLift Servers、接收遊戲工作階段的指示、報告狀態和運作狀態，以及執行其他動作。

本主題提供範例程式碼，新增使用 託管遊戲所需的最低功能 Amazon GameLift Servers。

步驟 1：更新GameMode標頭檔案

1. 開啟遊戲專案的程式碼檔案，並找到 *Your-application-name*GameMode.h 檔案。範例：GameLiftUnrealAppGameMode.h。如果您使用 Visual Studio，請開啟遊戲專案.sln的檔案。
2. 變更標頭檔案以包含下列範例程式碼。請務必將 "GameLiftUnrealApp" 取代為您自己的應用程式名稱。

GameMode.h 的範例程式碼

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

#pragma once

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

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

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

public:
    AGameLiftUnrealAppGameMode();

protected:
    virtual void BeginPlay() override;

private:
    void InitGameLift();

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

步驟 2：將必要的伺服器 SDK 呼叫新增至遊戲伺服器程式碼

使用本節中的範例程式碼來整合遊戲伺服器程式碼，以便與 搭配使用Amazon GameLift Servers。如需程式碼功能的詳細資訊，請參閱 [初始化伺服器程序](#)和 [適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x](#)。

Note

WITH_GAMELIFT 前置處理器旗標有兩個用途：

- 限制對 Unreal 伺服器建置的Amazon GameLift Servers後端 API 呼叫
- 確保不同 Unreal 建置目標之間的相容性

1. 開啟相關的來源 *Your-application-name*GameMode.cpp檔案。在我們的範例中：GameLiftUnrealAppGameMode.cpp。
2. 變更程式碼以符合下列範例程式碼。請務必將 "GameLiftUnrealApp" 的任何執行個體取代為您自己的應用程式名稱。

提供的程式碼範例示範如何新增與 整合所需的元素Amazon GameLift Servers。其中包含：

- 初始化 Amazon GameLift Servers API 用戶端。
- 實作回呼函數以回應來自 Amazon GameLift Servers服務的請求，包括 OnStartGameSession、OnProcessTerminate和 onHealthCheck。
- 準備好託管遊戲工作階段時，請呼叫 ProcessReady() 通知Amazon GameLift Servers服務。

GameMode.cpp 的範例程式碼

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

#include "GameLiftUnrealAppGameMode.h"

#include "UObject/ConstructorHelpers.h"
#include "Kismet/GameplayStatics.h"

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

```

#endif

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

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

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

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

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

#ifdef WITH_GAMELIFT
    InitGameLift();
#endif
}

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

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

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

    bool bIsAnywhereActive = false;

```

```

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

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

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

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

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

        FString glAnywhereHostId = "";

```



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

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

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

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

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

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

```

    }

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

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

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

```

```

}

ProcessParameters = MakeShared<FProcessParameters>();

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

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

```

```

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

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

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

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

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

    if (SwitchStr.Split("=", &Key, &Value))
    {
        if (Key.Equals("port"))
        {
            ProcessParameters->port = FString::Atoi(*Value);
        }
    }
}

```

```

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

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

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

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

```

步驟 3：重建遊戲專案

- 為下列兩種目標類型建置遊戲專案：Development Editor 和 Development Server。

 Note

您不需要重建解決方案。反之，請在應用程式的 /Games/ 資料夾下建置專案。否則，Visual Studio 會重建整個 UE5 專案，最多可能需要一小時。

封裝遊戲伺服器以進行託管

您的遊戲伺服器程式碼現在已與最基本的伺服器 SDK 功能整合，您已準備好使用 Unreal Editor 封裝遊戲伺服器組建。

封裝遊戲伺服器組建

1. 在 Unreal Editor 中開啟遊戲專案。
2. 遵循 Unreal Editor 步驟來封裝遊戲伺服器：
 - 選擇您的目標平台 (Windows 或 Linux)。
 - 選取您的伺服器建置目標 (*[your application name]*Server)。

封裝程序會產生您的遊戲伺服器可執行檔：*[your application name]*Server.exe。

3. 準備您的遊戲伺服器建置，以便部署以託管資源。組建應包含下列檔案：
 - 您的遊戲伺服器可執行檔
 - 如果您使用的是 Unreal Engine 5.5 版或更新版本，請針對 Windows 組建包含下列檔案。您可以在 Unreal Engine 的來源建置版本中找到它們：
 - VC_redist.x64.exe (UnrealEngine\Engine\Source\Programs\PrereqInstaller\Resources\VCRedist\)
 - UEPrereqSetup_x64.exe or UE5PrereqSetup_x64.exe (UnrealEngine\Engine\Extras\Redist\en-us\)
 - 遊戲伺服器的所有其他必要相依性。
 - 如有需要，OpenSSL 程式庫。如果您的遊戲伺服器已與Amazon GameLift Servers伺服器 SDK 5.3 版或更新版本整合，您可以略過此步驟。[您可以在此處取得最新的伺服器 SDK 版本。](#)

使用適用於 Unreal 的伺服器 SDK 5.2 版及更早版本建立的遊戲伺服器所需的 OpenSSL 程式庫

您必須包含與在 Unreal 中封裝遊戲伺服器時所使用的相同 OpenSSL 程式庫版本。這些程式庫位於您的遊戲引擎來源中。位置會根據您的開發環境而有所不同：

在 Windows 上：

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

在 Linux 上：

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

將 OpenSSL 程式庫複製到與遊戲伺服器可執行檔位於相同目錄中的遊戲組建套件。

後續步驟

現在您已使用託管所需的最低功能準備遊戲伺服器組建 Amazon GameLift Servers，請考慮下列可能的後續步驟：

- 為測試和開發部署整合的遊戲伺服器。使用 Anywhere 機群，您可以將本機機器設定為託管資源，並使用它來測試遊戲伺服器和遊戲用戶端連線。對於雲端託管，請將遊戲伺服器部署到受管 EC2 或受管容器機群。如需指引，請參閱這些主題：
 - [使用 Amazon GameLift Servers Anywhere 設定反覆開發](#)
 - [Amazon GameLift Servers Anywhere 機群](#)
 - [Amazon GameLift Servers 受管 EC2 機群](#)
 - [Amazon GameLift Servers 受管容器機群](#)
- 新增選用功能，自訂您的遊戲伺服器整合。例如，您可能想要新增具有唯一玩家 IDs 玩家工作階段、設定配對回填，或管理遊戲伺服器對其他 AWS 資源（例如資料庫或內容儲存服務）的存取。如需指引，請參閱這些主題：
 - [Amazon GameLift Servers 新增至您的遊戲伺服器](#)
 - [適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x](#)

- 自訂您的遊戲用戶端元件以請求遊戲工作階段、接收連線資訊，以及直接連線至遊戲伺服器以玩遊戲。如需指引，請參閱這些主題：
 - [將您的遊戲用戶端與 整合 Amazon GameLift Servers](#)

Amazon GameLift Servers 整合至 Unity 專案

了解如何將適用於 Unity 的 Amazon GameLift Servers SDK 整合到您的遊戲專案中，以存取完整的伺服器 SDK 功能集。

秘訣

若要加快部署速度，請嘗試 Unity Amazon GameLift Servers 的獨立外掛程式。它提供引導式 UI 工作流程，以最少的設定快速部署遊戲伺服器，因此您可以嘗試執行中的遊戲元件。請參閱 [Amazon GameLift ServersUnity 的外掛程式 \(伺服器 SDK 5.x\)](#)。

其他資源：

- [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)
- [the section called “取得開發工具”](#)

安裝適用於 Unity 的伺服器 SDK

從 [GitHub](#) 取得 Unity Amazon GameLift Servers 的開放原始碼。儲存庫的讀我檔案包含先決條件和安裝指示。

設定 Amazon GameLift ServersAnywhere 機群進行測試

您可以將開發工作站設定為 Amazon GameLift ServersAnywhere 託管機群，以反覆測試 Amazon GameLift Servers 整合。透過此設定，您可以在工作站上啟動遊戲伺服器程序、將玩家加入或配對請求傳送至 Amazon GameLift Servers 以啟動遊戲工作階段，並將用戶端連線至新的遊戲工作階段。透過將自己的工作站設定為託管伺服器，您可以監控與的遊戲整合的各個層面 Amazon GameLift Servers。

如需設定工作站的指示，請參閱 [使用 Amazon GameLift ServersAnywhere 設定本機測試](#) 以完成下列步驟：

1. 為您的工作站建立自訂位置。
2. 使用新的自訂位置建立 Amazon GameLift ServersAnywhere 機群。如果成功，此請求會傳回機群 ID。請記下此值，因為您稍後會需要它。

3. 在新的 Anywhere 機群中將工作站註冊為運算。提供唯一的運算名稱，並指定工作站的 IP 地址。如果成功，此請求會以 WebSocket URL 的形式傳回服務 SDK 端點。請記下此值，因為您稍後會需要它。
4. 為您的工作站運算產生身分驗證字符。此短期身分驗證包含權杖和過期日期。您的遊戲伺服器會使用它來驗證與 Amazon GameLift Servers 服務的通訊。將身分驗證存放在工作站運算上，以便您執行的遊戲伺服器程序可以存取。

將 Amazon GameLift Servers 伺服器程式碼新增至 Unity 專案

您的遊戲伺服器會與 Amazon GameLift Servers 服務通訊，以接收指示並報告持續狀態。若要達成此目的，您可以新增使用伺服器 SDK 的遊戲 Amazon GameLift Servers 伺服器程式碼。

提供的程式碼範例說明基本的必要整合元素。它使用 `MonoBehavior` 來說明使用 進行簡單的遊戲伺服器初始化 Amazon GameLift Servers。此範例假設遊戲伺服器在 Amazon GameLift Servers Anywhere 機群上執行以進行測試。它包含程式碼以：

- 初始化 Amazon GameLift Servers API 用戶端。此範例使用 `版本InitSDK()` 搭配 Anywhere 機群和運算的伺服器參數。使用 WebSocket URL、機群 ID、運算名稱（主機 ID）和身分驗證字符，如上一個主題 所定義 [設定 Amazon GameLift Servers Anywhere 機群進行測試](#)。
- 實作回呼函數來回應來自 Amazon GameLift Servers 服務的請求，包括 `OnStartGameSession`、`OnProcessTerminate` 和 `onHealthCheck`。
- 使用指定的連接埠呼叫 `ProcessReady()`，以在程序準備好託管遊戲工作階段時通知 Amazon GameLift Servers 服務。

提供的範例程式碼會建立與服務的通訊 Amazon GameLift Servers。它也會實作一組回呼函數，以回應來自 Amazon GameLift Servers 服務的請求。如需每個函數和程式碼功能的詳細資訊，請參閱 [初始化伺服器程序](#)。如需此程式碼中使用的 SDK 動作和資料類型的詳細資訊，請參閱 [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)。

範例程式碼示範如何新增所需的功能，如 [Amazon GameLift Servers 新增至遊戲伺服器](#) 中所述。如需伺服器 SDK 動作的詳細資訊，請參閱 [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)。

整合範例程式碼

```
using System.Collections.Generic;
using Aws.GameLift.Server;
using UnityEngine;
```

```
public class ServerSDKManualTest : MonoBehaviour
{
    //This example is a simple integration that initializes a game server process
    //that is running on an Amazon GameLift Servers Anywhere fleet.
    void Start()
    {
        //Identify port number (hard coded here for simplicity) the game server is
        //listening on for player connections
        var listeningPort = 7777;

        //WebSocketUrl from RegisterHost call
        var websocketUrl = "wss://us-west-2.api.amazongamelift.com";

        //Unique identifier for this process
        var processId = "myProcess";

        //Unique identifier for your host that this process belongs to
        var hostId = "myHost";

        //Unique identifier for your fleet that this host belongs to
        var fleetId = "myFleet";

        //Authorization token for this host process
        var authToken = "myAuthToken";

        //Server parameters are required for an Amazon GameLift Servers Anywhere fleet.
        //They are not required for an Amazon GameLift Servers managed EC2 fleet.
        ServerParameters serverParameters = new ServerParameters(
            websocketUrl,
            processId,
            hostId,
            fleetId,
            authToken);

        //InitSDK establishes a local connection with an Amazon GameLift Servers agent
        //to enable further communication.
        var initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
        if (initSDKOutcome.Success)
        {
            //Implement callback functions
            ProcessParameters processParameters = new ProcessParameters(
                //Implement OnStartGameSession callback
                (gameSession) => {
```

```

        //Amazon GameLift Servers sends a game session activation request
to the game server
        //with game session object containing game properties and other
settings.
        //Here is where a game server takes action based on the game
session object.
        //When the game server is ready to receive incoming player
connections,
        //it invokes the server SDK call ActivateGameSession().
        GameLiftServerAPI.ActivateGameSession();
    },
    (updateGameSession) => {
        //Amazon GameLift Servers sends a request when a game session is
updated (such as for
        //FlexMatch backfill) with an updated game session object.
        //The game server can examine matchmakerData and handle new
incoming players.
        //updateReason explains the purpose of the update.
    },
    () => {
        //Implement callback function OnProcessTerminate
        //Amazon GameLift Servers invokes this callback before shutting
down the instance hosting this game server.
        //It gives the game server a chance to save its state, communicate
with services, etc.,
        //and initiate shut down. When the game server is ready to shut
down, it invokes the
        //server SDK call ProcessEnding() to tell Amazon GameLift Servers
it is shutting down.
        GameLiftServerAPI.ProcessEnding();
    },
    () => {
        //Implement callback function OnHealthCheck
        //Amazon GameLift Servers invokes this callback approximately every
60 seconds.
        //A game server might want to check the health of dependencies,
etc.
        //Then it returns health status true if healthy, false otherwise.
        //The game server must respond within 60 seconds, or Amazon
GameLift Servers records 'false'.
        //In this example, the game server always reports healthy.
        return true;
    },

```

```

        //The game server gets ready to report that it is ready to host game
sessions
        //and that it will listen on port 7777 for incoming player connections.
        listeningPort,
        new LogParameters(new List<string>()
        {
            //Here, the game server tells Amazon GameLift Servers where to find
game session log files.
            //At the end of a game session, Amazon GameLift Servers uploads
everything in the specified
            //location and stores it in the cloud for access later.
            "/local/game/logs/myserver.log"
        }));

        //The game server calls ProcessReady() to tell Amazon GameLift Servers it's
ready to host game sessions.
        var processReadyOutcome =
GameLiftServerAPI.ProcessReady(processParameters);
        if (processReadyOutcome.Success)
        {
            print("ProcessReady success.");
        }
        else
        {
            print("ProcessReady failure : " +
processReadyOutcome.Error.ToString());
        }
    }
    else
    {
        print("InitSDK failure : " + initSDKOutcome.Error.ToString());
    }
}

void OnApplicationQuit()
{
    //Make sure to call GameLiftServerAPI.ProcessEnding() and
GameLiftServerAPI.Destroy() before terminating the server process.
    //These actions notify Amazon GameLift Servers that the process is terminating
and frees the API client from memory.
    GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();
    GameLiftServerAPI.Destroy();
    if (processEndingOutcome.Success)
    {

```

```
        Environment.Exit(0);
    }
    else
    {
        Console.WriteLine("ProcessEnding() failed. Error: " +
processEndingOutcome.Error.ToString());
        Environment.Exit(-1);
    }
}
}
```

後續步驟

現在您已使用 託管所需的最低功能準備遊戲伺服器組建Amazon GameLift Servers，請考慮下列可能的後續步驟：

- 為 和測試和開發部署整合的遊戲伺服器。使用 Anywhere 機群，您可以將本機機器設定為託管資源，並使用它來測試遊戲伺服器和遊戲用戶端連線。對於雲端託管，請將遊戲伺服器部署到受管 EC2 或受管容器機群。如需指引，請參閱這些主題：
 - [使用 Amazon GameLift Servers Anywhere 設定反覆開發](#)
 - [Amazon GameLift Servers Anywhere 機群](#)
 - [Amazon GameLift Servers 受管 EC2 機群](#)
 - [Amazon GameLift Servers 受管容器機群](#)
- 新增選用功能，自訂您的遊戲伺服器整合。例如，您可能想要新增具有唯一玩家 IDs 玩家工作階段、設定配對回填，或管理遊戲伺服器對其他 AWS 資源（例如資料庫或內容儲存服務）的存取。如需指引，請參閱這些主題：
 - [Amazon GameLift Servers 新增至您的遊戲伺服器](#)
 - [適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x](#)
- 自訂您的遊戲用戶端元件以請求遊戲工作階段、接收連線資訊，並直接連線至遊戲伺服器以玩遊戲。如需指引，請參閱這些主題：
 - [將您的遊戲用戶端與 整合 Amazon GameLift Servers](#)

設計您的遊戲用戶端服務

我們建議您實作遊戲用戶端服務，以驗證您的玩家並與 Amazon GameLift Servers API 通訊。透過實作自訂遊戲用戶端服務，您可以：

- 為您的玩家自訂身分驗證。
- 控制 如何Amazon GameLift Servers比對和啟動遊戲工作階段。
- 將玩家資料庫用於玩家屬性，例如配對的技能評分，而不是信任用戶端。

使用遊戲用戶端服務也可降低遊戲用戶端直接與您的 Amazon GameLift Servers API 互動所帶來的安全風險。

驗證您的玩家

您可以使用 Amazon Cognito 和玩家工作階段 IDs 來驗證您的遊戲用戶端。若要管理玩家身分的生命週期和屬性，請使用 Amazon Cognito 使用者集區。

如果您願意，請建置自訂身分解決方案並將其託管 AWS。您也可以使用 Lambda 授權方搭配 API Gateway 進行自訂授權邏輯。

其他資源：

- [使用身分集區（聯合身分）](#) (Amazon Cognito 開發人員指南)
- [使用者集區入門](#) (Amazon Cognito 開發人員指南)
- [如何使用 Amazon Cognito 設定玩家身分驗證](#) (AWS 適用於遊戲部落格)

具有無伺服器後端的獨立遊戲工作階段伺服器

使用無伺服器用戶端服務架構，後端可以從高度可擴展的資料庫檢視配對票證的狀態，而不是直接存取 Amazon GameLift Servers API。

下圖顯示使用 建置的無伺服器後端 AWS 服務，將玩家配對至在Amazon GameLift Servers機群上執行的遊戲。下列清單提供圖表中每個編號標註的說明。若要嘗試此範例，請參閱 GitHub 上的 [上的多玩家工作階段型遊戲託管 AWS](#)。

1. 遊戲用戶端會從 Amazon Cognito 身分集區請求 Amazon Cognito 使用者身分。
2. 遊戲用戶端會收到臨時存取憑證，並透過 Amazon API Gateway API 請求遊戲工作階段。
3. API Gateway 會叫用 AWS Lambda 函數。
4. Lambda 函數會從 Amazon DynamoDB NoSQL 資料表請求玩家資料。函數會在請求內容資料中提供 Amazon Cognito 身分。

5. Lambda 函數透過Amazon GameLift ServersFlexMatch配對來請求配對。
6. FlexMatch 會比對一組具有適當延遲的玩家，然後請求透過Amazon GameLift Servers佇列放置遊戲工作階段。佇列中具有一或多個 AWS 區域 位置的機群。
7. 在 將工作階段Amazon GameLift Servers放置在機群的其中一個位置之後， Amazon GameLift Servers 會將事件通知傳送至 Amazon Simple Notification Service (Amazon SNS) 主題。
8. Lambda 函數會接收並處理 Amazon SNS 事件。
9. 如果配對票證是MatchmakingSucceeded事件，則 Lambda 函數會將結果以及遊戲伺服器的連接埠和 IP 地址寫入 DynamoDB 資料表。
10. 遊戲用戶端向 API Gateway 提出簽署請求，以在特定間隔檢視配對票證的狀態。
11. API Gateway 使用 Lambda 函數來檢查配對票證狀態。
12. Lambda 函數會檢查 DynamoDB 資料表，以查看票證是否成功。如果成功，函數會將遊戲伺服器的連接埠和 IP 地址，以及玩家工作階段 ID 傳回用戶端。如果票證未成功，函數會傳送回應，確認相符項目尚未就緒。
13. 遊戲用戶端會使用後端服務提供的連接埠和 IP 地址，使用 TCP 或 UDP 連線至遊戲伺服器。遊戲用戶端接著會將玩家工作階段 ID 傳送至遊戲伺服器，然後使用適用於 的伺服器 SDK 驗證 IDAmazon GameLift Servers。

具有 WebSocket 型後端的獨立遊戲工作階段伺服器

使用 Amazon API Gateway WebSocket 架構，您可以使用 WebSockets 提出配對請求，並使用伺服器啟動的訊息傳送配對完成的推送通知。此架構透過在用戶端與伺服器之間進行雙向通訊來改善效能。

如需使用 API Gateway WebSock APIs的詳細資訊，請參閱[使用 WebSocket APIs](#)。

下圖顯示 WebSocket 型後端架構，該架構使用 API Gateway 和其他 AWS 服務 將玩家配對至在 Amazon GameLift Servers機群上執行的遊戲。下列清單提供圖表中每個編號標註的說明。

1. 遊戲用戶端會從 Amazon Cognito 身分集區請求 Amazon Cognito 使用者身分。
2. 遊戲用戶端使用 Amazon Cognito 登入資料簽署與 API Gateway API 的 WebSocket 連線。
3. API Gateway 會在連線上呼叫 AWS Lambda 函數。函數會將連線資訊存放在 Amazon DynamoDB 資料表中。
4. 遊戲用戶端會透過 WebSocket 連線的 API Gateway API，將訊息傳送至 Lambda 函數，以請求工作階段。
5. Lambda 函數會收到訊息，然後透過Amazon GameLift ServersFlexMatch配對請求配對。

6. 在 FlexMatch 符合一組玩家之後，會透過 Amazon GameLift Servers 佇列 FlexMatch 請求遊戲工作階段放置。
7. 在將工作階段 Amazon GameLift Servers 放置在機群的其中一個位置之後，Amazon GameLift Servers 會將事件通知傳送至 Amazon Simple Notification Service (Amazon SNS) 主題。
8. Lambda 函數會接收並處理 Amazon SNS 事件。
9. 如果配對票證是 MatchmakingSucceeded 事件，則 Lambda 函數會從 DynamoDB 請求正確的玩家可以連線。然後，函數會透過 WebSocket 連線的 API Gateway API，將訊息傳送至遊戲用戶端。在此架構中，遊戲用戶端不會主動輪詢配對的狀態。
10. 遊戲用戶端會透過 WebSocket 連線，接收遊戲伺服器的連接埠和 IP 地址，以及玩家工作階段 ID。
11. 遊戲用戶端會使用後端服務提供的連接埠和 IP 地址，使用 TCP 或 UDP 連線至遊戲伺服器。遊戲用戶端也會將玩家工作階段 ID 傳送至遊戲伺服器，然後使用適用於的伺服器 SDK 驗證 ID Amazon GameLift Servers。

使用 Amazon GameLift Servers Anywhere 設定反覆開發

Amazon GameLift Servers 提供工具和解決方案，協助您設定託管測試環境，以便在遊戲開發期間使用。透過這些工具，您可以建立環境，以模擬受管託管的真實世界玩家體驗，Amazon GameLift Servers 並支援快速、反覆的開發程序。

使用單獨的測試環境，您可以在測試期間移除 Amazon GameLift Servers 受管機群的負荷。您不再需要上傳每個新的遊戲伺服器建置反覆運算、為其建立新機群，然後等待 15 分鐘以上才能啟用。反之，您可以建立新的組建、使用新組建快速更新測試機群、啟動測試機群，並開始測試。

使用 Amazon GameLift Servers Anywhere 機群，您可以使用本機裝置設定測試環境，例如您的開發工作站。您也可以使用雲端託管資源來設定測試環境。

設定 Anywhere 測試環境來開發和測試各種案例，包括：

- 測試遊戲伺服器與 Amazon GameLift Servers 伺服器 SDK 的整合。即使沒有運作中的遊戲用戶端，您也可以使用 CLI AWS 呼叫來開始新的遊戲工作階段並追蹤遊戲工作階段事件，以進行測試。
- 在您為遊戲開發元件時，測試遊戲用戶端、後端服務和 Amazon GameLift Servers 服務之間的互動。微調玩家加入遊戲的體驗。
- 試用您的 FlexMatch 配對建構器設計。嘗試規則集變化和其他配對功能實作。設定和測試配對回填。
- 試用其他 Amazon GameLift Servers 託管功能，例如遊戲伺服器生命週期管理的執行期組態設定（使用 Amazon GameLift Servers 代理程式）。
- 在即時託管環境中快速建置、測試和重複，以驗證遊戲玩家體驗的所有層面，包括多玩家互動。

稍後，當您準備啟動遊戲時，您會想要新增Amazon GameLift Servers受管機群來微調託管組態，並測試其他案例，包括下列項目：

- 試用和測試遊戲工作階段佇列設計，包括使用多位置機群、Spot 和隨需機群，以及多種執行個體類型。
- 嘗試使用受管機群放置遊戲工作階段選項，包括使用選用的延遲政策和機群優先順序設定。
- 使用自動或手動擴展選項，設定容量擴展以符合玩家需求。
- AWS CloudFormation 使用Amazon GameLift Servers受管機群設定，以長期管理您的託管資源。

Fast Build Update Tool（僅供開發）

使用受管 EC2 機群，若要部署遊戲伺服器建置更新，您需要將每個新建置上傳至，Amazon GameLift Servers並為其建立新的機群。

Fast Build Update Tool 可讓您在開發期間略過這些步驟，節省時間並加快開發反覆運算速度。使用此工具，您可以快速更新現有機群中所有運算的遊戲建置檔案。工具有數個選項；您可以取代整個遊戲組建或變更 6 個特定檔案，也可以管理如何在更新後重新啟動遊戲伺服器程序。您也可以使用它來更新機群中的個別運算。

若要取得快速建置更新工具並進一步了解如何使用，請造訪 Github [中快速建置更新工具](#) Amazon GameLift Servers的工具組儲存庫。

主題

- [建置以雲端為基礎的測試環境](#)
- [使用 Amazon GameLift ServersAnywhere 設定本機測試](#)
- [使用 Amazon GameLift Servers 代理程式](#)

建置以雲端為基礎的測試環境

Note

本主題涵蓋與適用於 5.x Amazon GameLift Servers版的伺服器 SDK 整合的遊戲反覆測試。如果您的遊戲使用伺服器 SDK 4.x 版或更早版本，請參閱 [使用 Amazon GameLift Servers Local 測試您的整合](#)。

使用 Amazon GameLift ServersAnywhere 機群在雲端託管環境中反覆建置和測試遊戲元件。建立具有託管資源和 Amazon GameLift Servers 服務連線的 Anywhere 機群、在它們上執行遊戲伺服器，並視需要測試遊戲功能。

使用 Amazon GameLift Servers 代理程式部署 Anywhere 機群

如果您的遊戲伺服器組建與 Amazon GameLift Servers SDK 5.x 或更新版本整合，您可以使用 Amazon GameLift Servers 代理程式將其部署到雲端型 Anywhere 機群。代理程式是一種背景程序，可管理機群中每個運算的遊戲伺服器生命週期和其他任務。這些任務包括向 Anywhere 機群註冊運算、取得身分驗證字符，以及根據一組指示啟動/停止遊戲伺服器程序。代理程式是由機群的執行時間組態所控制，您可以在機群生命週期內隨時更新。（代理程式會自動部署到受管 EC2 機群。）如需詳細資訊和下載 代理程式，請參閱 [Amazon GameLift Servers GitHub 儲存庫](#)。

使用 Amazon EC2 設定反覆測試

使用此 [Amazon GameLift Servers 工具組解決方案](#) 中的引導式工作流程來設定以雲端為基礎的託管環境，以鏡射的受管託管體驗 Amazon GameLift Servers。

GitHub 儲存庫提供一組指令碼，可自動化大部分使用 Amazon GameLift ServersAnywhere 和 Amazon GameLift Servers Agent 設定測試環境的程序。它也提供每當您有新的遊戲伺服器建置要測試時更新環境的指引。您可以使用範例遊戲伺服器建置執行單一指令碼來部署測試環境，也可以逐步演練每個步驟，以使用您自己的遊戲伺服器建置進行設定。

在此工作流程中，您將完全在 中運作 AWS Management Console，使用 AWS CloudShell 執行指令碼和完成命令列任務。

Note

對於本教學課程中的任務，您需要具有下列服務許可 AWS 的帳戶使用者：Amazon GameLift Servers、AWS CloudShell、Amazon S3 AWS Systems Manager、Amazon EC2 和 AWS Identity and Access Management。具有 AWS 帳戶管理員層級存取權的使用者已經擁有必要的許可。

工作流程涵蓋下列任務：

- 封裝適用於的遊戲伺服器組建 Amazon GameLift Servers。工作流程提供指令碼來建置範例 C++ 遊戲伺服器，該伺服器已與適用於 5.x Amazon GameLift Servers 版的伺服器 SDK 整合，並準備好託管。或者，如果您已完成整合，則可以使用自己的遊戲專案。

- 設定 Amazon Simple Storage Service 儲存貯體來存放遊戲伺服器建置和相依性。當您產生遊戲組建的新版本時，您可以將它們存放在 S3 中，並使用指令碼來更新 Anywhere 機群以進行遊戲測試。
- 取得並建置 Amazon GameLift Servers 代理程式。代理程式會根據您的組態，管理託管資源上的遊戲伺服器程序。它使用相同的邏輯，並且與 Amazon GameLift Servers 受管 EC2 託管的行為相同。
- 為您的託管資源設定 Anywhere 機群。透過 Anywhere 機群，您可以使用 Amazon GameLift Servers 服務來託管非管理的資源 Amazon GameLift Servers。在此步驟中，您也將設定執行時間組態，指示 Amazon GameLift Servers 代理程式何時及如何啟動遊戲伺服器程序。
- 設定 Amazon EC2 執行個體。這是用於反覆測試的測試環境。使用標準 EC2 執行個體，而不是全受管 Amazon GameLift Servers 執行個體（已針對生產層級使用進行最佳化）的速度快得多。使用標準 EC2 執行個體，您可以視需要快速並持續更新遊戲伺服器。
- 將遊戲伺服器建置和 Amazon GameLift Servers 代理程式部署至 Amazon EC2 執行個體。工作流程提供的指令碼可取得遊戲組建的最新版本，以及所有相依性，並將其安裝在 EC2 執行個體上。在此工作流程中，相依性包括 Amazon GameLift Servers Agent 和 CloudWatch Agent。
- 啟動 Amazon GameLift Servers 代理程式。安裝後，代理程式會自動啟動並開始執行指示。其中包含：
 - 在 Amazon GameLift Servers Anywhere 機群中將 EC2 執行個體註冊為運算。
 - 建立與 Amazon GameLift Servers 服務的 WebSocket 連線，並取得最新的執行時間組態。
 - 根據執行時間組態中的指示啟動遊戲伺服器程序。在此工作流程中，代理程式會指示啟動遊戲伺服器可執行檔的單一程序。
- 測試您的遊戲案例。設定測試環境並安裝最新的遊戲伺服器建置後，您就可以開始測試。工作流程會逐步執行數個測試步驟，包括啟動遊戲工作階段。存取 CloudWatch 遊戲伺服器日誌，以在遊戲工作階段啟動時追蹤進度，並準備接受玩家。

當您開發遊戲元件時，包括遊戲用戶端和用戶端後端服務，您可以將這些元件包含在測試案例中。使用遊戲用戶端請求遊戲工作階段、從 Amazon GameLift Servers 服務擷取連線資訊，然後直接連線至遊戲工作階段。

- 部署新的遊戲伺服器建置並重複測試。當您開發遊戲時，您可以產生新的遊戲伺服器組建，然後快速將其部署到 EC2 測試環境進行測試。將它們上傳至 Amazon S3 儲存貯體，然後使用工作流程指令碼來更新測試環境。

將您的遊戲轉換為 Amazon GameLift Servers 受管機群

完成開發測試並準備好準備啟動後，這是切換到 Amazon GameLift Servers 受管機群的好時機。使用受管機群來微調和測試您的遊戲託管資源。實作遊戲工作階段置放解決方案（佇列和配對建構器），選

取最佳託管硬體（包括 Spot 機群）和位置，然後選擇擴展容量的策略。您可能也想要開始使用 AWS CloudFormation 來更有效率地管理所有遊戲託管資源的生命週期，包括機群、佇列和配對建構器。

從雲端型 Anywhere 測試機群轉換到 Amazon GameLift Servers 受管機群需要最少的精力。您不需要變更任何遊戲程式碼，而且可以重複使用相同的佇列和配對建構器。執行下列任務：

- 建立 Amazon GameLift Servers 建置資源。使用 Anywhere 測試機群，您必須手動將遊戲伺服器建置和相依性部署到每個機群運算。使用受管機群，將您的遊戲建置套件上傳至 Amazon GameLift Servers，以自動將其部署至所有機群運算。[部署用於 Amazon GameLift Servers 託管的自訂伺服器建置](#) 如需封裝遊戲建置檔案，以及使用 Amazon S3 儲存貯體中的檔案建立建置資源的詳細資訊，請參閱。
- 建立受管機群。使用主控台或 AWS CLI 來建立機群，指定 EC2 受管機群。這種類型的機群需要額外的組態設定，包括指定建置資源和執行個體類型。您可以使用相同的執行時間組態來管理每個機群運算的遊戲伺服器生命週期。[建立 Amazon GameLift Servers 受管 EC2 機群](#) 如需建立受管機群的詳細資訊，請參閱。
- 重新導向機群別名（選用）。如果您將別名設定為與 Anywhere 機群搭配使用，則可以為受管機群重複使用相同的別名。[建立 Amazon GameLift Servers 別名](#) 如需建立或更新別名的詳細資訊，請參閱。

使用 Amazon GameLift Servers Anywhere 設定本機測試

Note

本主題涵蓋與適用於 5.x Amazon GameLift Servers 版的伺服器 SDK 整合之遊戲的本機測試。如果您的遊戲使用伺服器 SDK 4.x 版或更早版本，請參閱 [使用 Amazon GameLift Servers Local 測試您的整合](#)。

使用 Amazon GameLift Servers Anywhere 機群和您自己的硬體，在模擬託管環境中反覆建置和測試遊戲元件。設定 Anywhere 機群並註冊本機裝置，以建立 Amazon GameLift Servers 服務的連線。在裝置上安裝遊戲伺服器組建、啟動遊戲伺服器程序，並視需要測試遊戲功能。您可以視需要更新遊戲伺服器組建，以測試每個新的組建反覆運算。

透過 Anywhere 機群，您可以使用 AWS CLI 或測試指令碼進行測試。如果您已將遊戲用戶端與整合 Amazon GameLift Servers，您可以在相同的本機裝置上或不同的裝置上執行用戶端。

使用 Anywhere 機群在本機測試對於測試與的遊戲伺服器整合特別有用 Amazon GameLift Servers。您可以完整了解本機電腦上的所有託管活動，以及事件和記錄資料。

 Note

您是否使用 Unreal Engine 或 Unity 的 Amazon GameLift Servers 外掛程式？這些工具包括使用 Anywhere 機群設定本機測試的引導式工作流程。遵循 [Unity 的外掛程式：使用 Amazon GameLift Servers Anywhere 設定本機測試](#) 或的文件 [Unreal 的外掛程式：使用 Amazon GameLift Servers Anywhere 在本機託管您的遊戲](#)。

主題

- [設定 local Anywhere 機群](#)
- [更新並安裝遊戲伺服器](#)
- [測試遊戲工作階段活動](#)
- [在您的遊戲伺服器上反覆運算](#)
- [將您的遊戲轉換為 Amazon GameLift Servers 受管機群](#)

設定 local Anywhere 機群

請依照下列步驟，為您的本機工作站建立 Anywhere 機群。如需使用 AWS CLI 或 AWS Management Console 適用於的詳細說明 Amazon GameLift Servers，請參閱 [建立 Amazon GameLift Servers Anywhere 機群](#)。

建立 Anywhere 機群

1. 為您的本機工作站建立自訂位置。(AWS CLI 或主控台)。自訂位置只是您計劃包含在 Anywhere 機群中的運算資源的標籤。自訂位置名稱必須以開頭 custom-。例如：custom-my_laptop。請參閱 [建立自訂位置](#)。
2. 建立 Anywhere 機群 (AWS CLI 或主控台)。在此步驟中，使用本機工作站的自訂位置建立機群資源。請參閱 [建立 Anywhere 機群](#)。

請記下新機群的 ID 或 ARN 值。在下一個步驟中，您將需要用到此值。

3. 將本機工作站註冊為機群運算（僅限 AWS CLI）。Anywhere 機群必須至少有一個運算資源來託管您的遊戲伺服器。請參閱 [將運算新增至機群](#)。若要將運算新增至機群，您需要下列資訊：
 - 運算名稱。機群中的每個運算都必須具有唯一的名稱。
 - Anywhere 機群識別符。您可以使用 FleetID 或 FleetArn。

- 運算的連線資訊。指定 `IpAddress` 或 `DnsName`。這是 Amazon GameLift Servers 和 遊戲用戶端連線到遊戲伺服器的方式。
- Anywhere 機群中的自訂位置。

記下 `GameLiftServiceSdkEndpoint` 傳回值。當您更新遊戲伺服器以在 Anywhere 機群上執行時，將需要此值。

更新並安裝遊戲伺服器

此任務假設您已將遊戲伺服器建置與 Amazon GameLift Servers 伺服器 SDK 5.x 整合。整合程序涉及將程式碼新增至遊戲伺服器，使其可以與 Amazon GameLift Servers 服務互動，以啟動和管理遊戲工作階段。

對於 Anywhere 機群，您需要手動設定特定遊戲伺服器設定。在 Amazon GameLift Servers 受管機群上，這些設定會自動設定。

為 Anywhere 機群準備遊戲伺服器

1. 取得身分驗證字符。您的遊戲伺服器必須在與服務的每次通訊中包含身分驗證字符 Amazon GameLift Servers。Amazon GameLift Servers 身分驗證字符是短期的，必須定期重新整理。

最佳實務是建立指令碼以完成下列任務：

- 呼叫 AWS CLI 動作 `get-compute-auth-token`。
- 將傳回的字符值存放在遊戲伺服器程序可以擷取的位置，例如本機運算的環境變數中。

在運算中搭配遊戲伺服器安裝指令碼。將指令碼設定為在開始第一個遊戲伺服器程序之前執行。當遊戲伺服器程序處於作用中狀態時，請定期執行指令碼以維護有效的身分驗證字符。運算上的所有遊戲伺服器程序都可以使用相同的身分驗證字符。

2. 更新您的 Amazon GameLift Servers 遊戲伺服器程式碼。當您將遊戲伺服器程式碼與適用於的伺服器 SDK 整合時 Amazon GameLift Servers，您會將呼叫新增至動作 `InitSdk()`。當遊戲伺服器在 Anywhere 機群上執行時，此呼叫需要額外的伺服器參數。如需詳細資訊，請參閱 [初始化伺服器程序](#) 和您的開發語言 [適用於的伺服器 SDK 5.x Amazon GameLift Servers](#) 的。伺服器參數為：

- `websocketUrl` – 將此參數設定為當您向機群註冊運算時傳回 `GameLiftServiceSdkEndpoint` 的值。
- `hostId` – 將此參數設定為您在向 Anywhere 機群註冊運算時指定的運算名稱。

- `fleetId` – 將此參數設定為 Anywhere 機群的 ID。
- `authToken` – 將此參數設定為權杖，此權杖會傳回以回應擷取運算真我權杖的請求。
- `processId` – 設定此參數以識別在本機運算上執行的遊戲伺服器程序。每個並行遊戲伺服器程序都必須具有唯一的程序 ID。

每個遊戲伺服器程序使用的伺服器參數值，必須專屬於執行程序的 Anywhere 機群運算。如需如何取得運算適當值的詳細資訊，請參閱 [將運算新增至機群](#)。最佳實務是在本機運算上將 `webSocketUrl`、`fleetId`、`hostId` 和 `authToken` 設定為環境變數。在運算上執行的所有伺服器程序都會使用這些值。

3. 在本機運算上安裝遊戲伺服器組建。包含執行遊戲伺服器所需的所有相依性。
4. 啟動在本機運算上執行的一或多個遊戲伺服器程序。當遊戲伺服器程序呼叫伺服器 SDK 動作 `ProcessReady()`，程序已準備好託管遊戲工作階段。

測試遊戲工作階段活動

使用遊戲工作階段測試您的遊戲伺服器整合。如果您沒有與 Amazon GameLift Servers 功能整合的遊戲用戶端，您可以使用 AWS CLI 來啟動遊戲工作階段。嘗試下列案例：

- 建立遊戲工作階段。呼叫 [create-game-session](#) 命令（或 [CreateGameSession](#) API 操作）。指定您的 Anywhere 機群 ID 和自訂位置。此呼叫會傳回新遊戲工作階段的唯一識別符。
- 檢查遊戲工作階段狀態。呼叫 [describe-game-sessions](#) 命令（或 [DescribeGameSessions](#) API 動作）。指定遊戲工作階段 ID。此呼叫會傳回詳細的遊戲工作階段資訊，包括遊戲工作階段狀態。處於作用中狀態的遊戲工作階段已準備好供玩家連線。若要取得機群的所有遊戲工作階段清單，請呼叫 [list-game-sessions](#) 命令（或 [ListGameSessions](#) API 動作）。
- 連線至遊戲工作階段。如果您的遊戲用戶端能夠加入遊戲工作階段，請使用遊戲工作階段資訊中包含的連線資訊。

在您的遊戲伺服器上反覆運算

您可以使用相同的 Anywhere 機群和運算來測試遊戲伺服器組建的其他版本。

1. 清除現有的 **GameSession**。如果遊戲伺服器程序當機或無法呼叫 `ProcessEnding()`，會在遊戲伺服器停止傳送運作狀態檢查 `GameSession` 後 Amazon GameLift Servers 清除。
2. 產生新的遊戲伺服器組建。變更遊戲伺服器並封裝修訂後的組建。

3. 在本機運算上更新遊戲伺服器組建。您先前的 Anywhere 機群仍處於作用中狀態，且您的筆記型電腦仍註冊為機群中的運算資源。
4. 取得更新的授權字符。呼叫 [get-compute-auth-token](#) CLI 命令，並將權杖存放在本機運算上。
5. 啟動在本機運算上執行的一或多個遊戲伺服器程序。當遊戲伺服器程序呼叫 `ProcessReady()`，即可用於測試。

將您的遊戲轉換為Amazon GameLift Servers受管機群

完成開發測試並準備好準備啟動後，這是切換到Amazon GameLift Servers受管機群的好時機。使用受管機群來微調和測試您的遊戲託管資源。實作遊戲工作階段置放解決方案（佇列和配對建構器），選取最佳託管硬體（包括 Spot 機群）和位置，然後選擇擴展容量的策略。您也可以開始使用 AWS CloudFormation，更有效地管理所有遊戲託管資源的生命週期，包括機群、佇列和配對建構器。

您需要稍微修改，才能從本機 Anywhere 測試機群轉換到Amazon GameLift Servers受管機群。您可以重複使用相同的佇列和配對建構器。執行下列任務：

- 將遊戲伺服器程式碼呼叫變更為 **InitSdk()**。移除伺服器參數。對於受管機群，Amazon GameLift Servers會自動追蹤此資訊。
- 建立Amazon GameLift Servers建置資源。使用 Anywhere 測試機群，您必須手動將遊戲伺服器建置和相依性部署到每個機群運算。使用受管機群，您可以建立遊戲建置套件並將其上傳至 Amazon GameLift Servers，以自動將其部署至所有機群運算。[部署用於Amazon GameLift Servers託管的自訂伺服器建置](#) 如需封裝遊戲建置檔案，以及使用 Amazon S3 儲存貯體中的檔案建立建置資源的詳細資訊，請參閱。請勿包含註冊運算並取得身分驗證字符的指令碼，因為 Amazon GameLift Servers 會自動使用受管機群處理這些任務。
- 建立受管機群。使用主控台或 AWS CLI 建立機群，指定 EC2 受管機群。這種類型的機群需要額外的組態設定，包括指定建置資源和執行個體類型。所有 都需要設定執行時間組態，以管理每個機群運算的遊戲伺服器生命週期。[建立Amazon GameLift Servers受管 EC2 機群](#) 如需建立受管機群的詳細資訊，請參閱。
- 重新導向機群別名（選用）。如果您將別名設定為與 Anywhere 機群搭配使用，則可以為受管機群重複使用相同的別名。[建立Amazon GameLift Servers別名](#) 如需建立或更新別名的詳細資訊，請參閱。

使用 Amazon GameLift Servers 代理程式

Amazon GameLift Servers 代理程式會監督您Amazon GameLift Servers機群上遊戲伺服器程序的執行。代理程式會部署到機群中的每個運算，並為運算提供自動化程序管理、託管管理和記錄。若要使用

代理程式，您必須將遊戲伺服器建置與 5.x Amazon GameLift Servers 版或更新版本的伺服器開發套件整合。

Amazon GameLift Servers 代理程式可在外部用於非受管 EC2 Amazon GameLift Servers 機群的機群。(受管 EC2 機群會自動處理代理程式的任務。) 您可以選擇執行機 Amazon GameLift Servers 群，包括 Anywhere 機群，無論是否有代理程式。如果沒有代理程式，您必須提供完成必要任務的替代解決方案。

部署到運算時，應在啟動任何遊戲伺服器程序之前啟動 Amazon GameLift Servers 代理程式。啟動時，代理程式會完成下列任務：

- 使用 [RegisterCompute](#) API 向 Amazon GameLift Servers Anywhere 機群註冊運算。
- 呼叫 [GetComputeAuthToken](#) API 來擷取授權字符，並存放它以供在運算上執行的伺服器程序使用。
- 設定運算的 WebSocket URL 環境變數，並建立與服務的 WebSocket 連線 Amazon GameLift Servers。
- 從 Amazon GameLift Servers 服務請求機群執行時間組態的最新版本。
- 根據執行時間組態指示啟動和停止伺服器程序。

Amazon GameLift Servers 客服人員的原始程式碼和建置說明可在 [Amazon GameLift Servers 客服人員](#) GitHub 中取得。

關於 代理程式

Amazon GameLift Servers 代理程式旨在為您的機群處理下列任務：

程序管理

- 啟動執行時間指示中定義的新伺服器程序。代理程式可能會使用隨代理程式一起部署的自訂執行期組態。或者，您可以提供 RuntimeConfiguration 做為機群定義的一部分。此方法在 中具有優勢，您可以隨時修改機群的執行時間組態。代理程式會定期向 Amazon GameLift Servers 服務請求更新的執行時間組態。
- 監控伺服器程序啟用，並在未及時啟用時終止程序。
- 將活動訊號傳送至 Amazon GameLift Servers。如果代理程式無法傳送活動訊號，則運算可能會標記為過時。
- Amazon GameLift Servers 當伺服器程序結束時向 報告。Amazon GameLift Servers 會使用此資訊來監控遊戲伺服器在遊戲工作階段置放時的可用性。
- 發出伺服器程序的機群事件，包括：

- `SERVER_PROCESS_INVALID_PATH`：遊戲伺服器程序啟動參數的設定不正確。
- `SERVER_PROCESS_TERMINATED_UNHEALTHY`：遊戲伺服器程序未在啟用後 3 分鐘內報告有效的運作狀態檢查，因此已終止。
- `SERVER_PROCESS_FORCE_TERMINATED`：遊戲伺服器程序在 30 秒內 `OnProcessTerminate()` 傳送後未完全結束。
- `SERVER_PROCESS_CRASHED`：遊戲伺服器程序因某些原因而當機。

運算管理

- 從 Amazon GameLift Servers 服務接收訊息以關閉運算。
- 提示要由終止的運算 Amazon GameLift Servers。

日誌

- 將日誌上傳至您 AWS 帳戶中的 Amazon S3 儲存貯體。

使用 Amazon GameLift Servers Local 測試您的整合

Note

本主題僅涵蓋與 4.x 版或更早 Amazon GameLift Servers 版本的伺服器 SDK 整合的遊戲測試。您的伺服器 SDK 套件包含相容版本的 Amazon GameLift Servers Local。如果您使用的是伺服器開發套件 5.x 版，請參閱 [使用 Amazon GameLift Servers Anywhere 設定本機測試](#) 以取得 Amazon GameLift Servers Anywhere 機群的本機測試。

透過 Amazon GameLift Servers Local，即可在本機裝置上執行受管 Amazon GameLift Servers 服務限制版本，並藉此測試遊戲整合。此工具在對遊戲整合進行重複性開發時非常有用。替代方案是將每個新組建上傳到 Amazon GameLift Servers 並設定機群來託管您的遊戲，每次都可能需要幾個或更多。

透過 Amazon GameLift Servers Local，您可以驗證下列項目：

- 您的遊戲伺服器與伺服器開發套件正確整合，並與 Amazon GameLift Servers 進行適當通訊，以啟動新遊戲工作階段、接受新玩家，以及報告運作狀況和狀態。
- 您的遊戲用戶端已正確與適用於的 AWS 開發套件整合，Amazon GameLift Servers 並且能夠擷取現有遊戲工作階段的資訊、開始新的遊戲工作階段、加入玩家至遊戲並連線至遊戲工作階段。

Amazon GameLift Servers Local 是命令列工具，可啟動受管 Amazon GameLift Servers 服務的獨立版本。Amazon GameLift Servers Local 也提供伺服器程序初始化、運作狀態檢查，以及 API 呼叫和回應的執行事件日誌。Amazon GameLift Servers Local 會辨識的部分 AWS SDK 動作 Amazon GameLift Servers。您可以從 AWS CLI 或您的遊戲用戶端進行呼叫。所有 API 動作都在本機執行，就像在 Amazon GameLift Servers web 服務中一樣。

每個伺服器程序應該只託管單一遊戲工作階段。遊戲工作階段是您用來連線至 Amazon GameLift Servers Local 的可執行檔。當遊戲工作階段完成時，您應該呼叫 `GameLiftServerSDK::ProcessEnding`，然後結束程序。使用 Amazon GameLift Servers Local 在本機測試時，您可以啟動多個伺服器程序。每個程序都會連線至 Amazon GameLift Servers Local。然後，您可以為每個伺服器程序建立一個遊戲工作階段。遊戲工作階段結束時，遊戲伺服器程序應該會結束。然後，您必須手動啟動另一個伺服器程序。

Amazon GameLift Servers local 支援下列 APIs：

- `CreateGameSession`
- `CreatePlayerSession`
- `CreatePlayerSessions`
- `DescribeGameSessions`
- `DescribePlayerSessions`

設定 Amazon GameLift Servers 本機

Amazon GameLift Servers Local 是搭配 `.jar` 伺服器開發套件的 [可執行](#) 檔案。它可以在 Windows 或 Linux 上執行，並搭配任何 Amazon GameLift Servers 支援的語言使用。

執行 Local 之前，您必須同時安裝下列項目。

- 適用於 3.1.5 版至 4.x Amazon GameLift Servers 版的伺服器 SDK 建置。
- Java 8

測試遊戲伺服器

如果您只想測試遊戲伺服器，您可以使用 AWS CLI 來模擬遊戲用戶端對 Amazon GameLift Servers Local 服務的呼叫。這會驗證您的遊戲伺服器是否依下列操作執行：

- 遊戲伺服器會正常啟動並初始化適用於的伺服器 SDK Amazon GameLift Servers。

- 做為啟動程序的一部分，遊戲伺服器會通知 Amazon GameLift Servers，伺服器已準備好託管遊戲工作階段。
- 遊戲伺服器在執行時每分鐘會向 Amazon GameLift Servers 傳送運作狀態。
- 遊戲伺服器回應請求以啟動新遊戲工作階段。

1. 啟動 Amazon GameLift Servers Local。

開啟命令提示字元視窗，導覽到 *GameLiftLocal.jar* 檔案所在的目錄並執行。在預設情況下，Local 會接聽連接埠 8080 上的遊戲用戶端請求。若要指定不同的連接埠號碼，請使用 *-p* 參數，如下例所示：

```
java -jar GameLiftLocal.jar -p 9080
```

一旦 Local 啟動，您會看到記錄顯示兩個本機伺服器已經啟動，一個接聽遊戲伺服器，一個接聽遊戲用戶端或 AWS CLI。記錄會持續報告兩個本機伺服器上的活動，包括與遊戲元件的往來通訊。

2. 啟動遊戲伺服器。

在本機啟動您的 Amazon GameLift Servers 整合遊戲伺服器。您不需要變更遊戲伺服器的端點。

在 Local 命令提示字元視窗中，記錄訊息表示您的遊戲伺服器已連接到 Amazon GameLift Servers Local 服務。這表示您的遊戲伺服器已成功初始化適用於的伺服器 SDK Amazon GameLift Servers (使用 *InitSDK()*)。它呼叫顯示記錄路徑的 *ProcessReady()*，如果成功，則準備託管遊戲工作階段。當遊戲伺服器執行時，Amazon GameLift Servers 會記錄遊戲伺服器的每個運作狀態報告。下列記錄訊息範例顯示已成功整合的遊戲伺服器：

```
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - SDK
connected: /127.0.0.1:64247
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - SDK pid is 17040,
sdkVersion is 3.1.5 and sdkLanguage is CSharp
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - NOTE: Only SDK
versions 3.1.5 and above are supported in GameLiftLocal!
16:50:53,451 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onProcessReady
received from: /127.0.0.1:64247 and ackRequest requested? true
16:50:53,543 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onProcessReady
data: logPathsToUpload: "C:\\game\\logs"
logPathsToUpload: "C:\\game\\error"
port: 1935
```



```
16:50:53,544 INFO || - [HostProcessManager] nioEventLoopGroup-3-1 - Registered new
process true, true,
16:50:53,558 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onReportHealth
received from /127.0.0.1:64247 with health status: healthy
```

可能發生的錯誤和警告訊息包括：

- 錯誤：「ProcessReady 找不到具有 pID 的程序：<process ID>！是否叫用 InitSDK()？」
- 警告：「使用 pID 的程序已存在程序狀態：<process ID>！ProcessReady(...) 是否多次叫用？」

3. 啟動 AWS CLI。

一旦遊戲伺服器成功呼叫 ProcessReady()，您就可以開始進行用戶端呼叫。開啟另一個命令提示字元視窗，然後啟動 AWS CLI 工具。AWS CLI 根據預設，會使用 Amazon GameLift Servers Web 服務端點。您必須使用 --endpoint-url 參數，在每個請求中以 Local 端點將此覆寫，如下列範例請求所示。

```
AWS gamelift describe-game-sessions --endpoint-url http://localhost:9080 --fleet-
id fleet-123
```

在 AWS CLI 命令提示字元視窗中，AWS gamelift 命令會產生回應，如 [AWS CLI 命令參考](#) 中所述。

4. 建立遊戲工作階段。

使用 AWS CLI 提交 [CreateGameSession\(\)](#) 請求。此請求應遵循預期的語法。對於 Local，此 FleetId 參數可設定為任何有效字串 (^fleet-\S+)。

```
AWS gamelift create-game-session --endpoint-url http://localhost:9080 --maximum-
player-session-count 2 --fleet-id
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
```

在 Local 命令提示字元視窗中，其所顯示的日誌訊息表示 Amazon GameLift Servers Local 已向遊戲伺服器傳送 onStartGameSession 回呼。如果成功建立遊戲工作階段，您的遊戲伺服器會透過叫用 ActivateGameSession 進行回應。

```
13:57:36,129 INFO || - [SDKInvokerImpl]
Thread-2 - Finished sending event to game server to start a game session:
```

```

arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-ab423a4b-b827-4765-
aea2-54b3fa0818b6.
    Waiting for ack response.13:57:36,143  INFO || - [SDKInvokerImpl]
    Thread-2 - Received ack response: true13:57:36,144  INFO || -
    [CreateGameSessionDispatcher] Thread-2 - GameSession with id:
    arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-ab423a4b-b827-4765-
aea2-54b3fa0818b6
    created13:57:36,227  INFO || - [SDKListenerImpl]
    nioEventLoopGroup-3-1 - onGameSessionActivate received
from: /127.0.0.1:60020 and ackRequest
    requested? true13:57:36,230  INFO || - [SDKListenerImpl]
    nioEventLoopGroup-3-1 - onGameSessionActivate data: gameSessionId:
    "arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890"

```

在 AWS CLI 視窗中，會以遊戲工作階段物件 Amazon GameLift Servers 回應，包括遊戲工作階段 ID。請注意，新遊戲工作階段的狀態為 **Activating** (啟動中)。一旦遊戲伺服器叫用 **ActivateGameSession**，此狀態會變更為 **Active** (作用中)。如果您想要查看已變更的狀態，請使用 AWS CLI 呼叫 **DescribeGameSessions()**。

```

{
  "GameSession": {
    "Status": "ACTIVATING",
    "MaximumPlayerSessionCount": 2,
    "FleetId": "fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d",
    "GameSessionId": "arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890",
    "IpAddress": "127.0.0.1",
    "Port": 1935
  }
}

```

測試遊戲伺服器和用戶端

若要檢查完整的遊戲整合，包括將玩家連接到遊戲，您可以在本機同時執行遊戲伺服器和用戶端。這可讓您測試遊戲用戶端到 Amazon GameLift Servers Local 的程式設計呼叫。您可以驗證下列動作：

- 遊戲用戶端成功向 Amazon GameLift Servers Local 服務提出 AWS SDK 請求，包括建立遊戲工作階段、擷取現有遊戲工作階段的資訊，以及建立玩家工作階段。
- 當玩家嘗試加入遊戲工作階段時，遊戲伺服器正確驗證玩家。對於已驗證的玩家，遊戲伺服器可能會擷取玩家資料 (如有實作)。
- 當玩家離開遊戲時，遊戲伺服器會報告連接中斷。
- 遊戲伺服器會報告結束遊戲工作階段。

1. 啟動 Amazon GameLift Servers Local。

開啟命令提示字元視窗，導覽到 *GameLiftLocal.jar* 檔案所在的目錄並執行。在預設情況下，Local 會接聽連接埠 8080 上的遊戲用戶端請求。若要指定不同的連接埠號碼，請使用 `-p` 參數，如下例所示。

```
./gamelift-local -p 9080
```

一旦 Local 啟動，您會看到記錄顯示兩個本機伺服器已經啟動，一個接聽遊戲伺服器，一個接聽遊戲用戶端或 AWS CLI。

2. 啟動遊戲伺服器。

在本機啟動您的 Amazon GameLift Servers 整合遊戲伺服器。如需更多訊息記錄的詳細資訊，請參閱 [測試遊戲伺服器](#)。

3. 設定 Local 遊戲用戶端並啟動。

若要將遊戲用戶端與 Amazon GameLift Servers Local 服務搭配使用，請務必針對遊戲用戶端的設定進行以下變更，如 [在後端服務 Amazon GameLift Servers 上設定](#) 中所述：

- 將 ClientConfiguration 物件變更為指向您的 Local 端點，例如 `http://localhost:9080`。
- 設定目標機群 ID 值。對於 Local，您不需要實際的機群 ID；請將目標機群設定為任何有效字串 (^fleet-\S+)，例如 `fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d`。
- 設定 AWS 登入資料。對於 Local，您不需要實際的 AWS 登入資料；您可以將存取金鑰和私密金鑰設定為任何字串。

在 Local 命令提示字元視窗中，一旦您啟動遊戲用戶端，其所顯示的日誌訊息應該會指出 GameLiftClient 已初始化，且成功與 Amazon GameLift Servers 服務進行通訊。

4. 測試遊戲用戶端對 Amazon GameLift Servers 服務的呼叫。

驗證您的遊戲用戶端是否成功進行任何或所有下列 API 呼叫：

- [CreateGameSession\(\)](#)
- [DescribeGameSessions\(\)](#)
- [CreatePlayerSession\(\)](#)
- [CreatePlayerSessions\(\)](#)
- [DescribePlayerSessions\(\)](#)

在 Local 命令提示字元視窗中，只有呼叫 `CreateGameSession()` 才會產生記錄訊息。若 Amazon GameLift Servers Local 提示遊戲伺服器啟動遊戲工作階段 (`onStartGameSession` 回呼)，並在遊戲伺服器進行叫用時取得 `ActivateGameSession` 成功狀態，日誌訊息將隨即顯示。在 AWS CLI 視窗中，所有 API 呼叫都會產生回應或錯誤訊息做為記錄。

5. 驗證您的遊戲伺服器正在驗證新玩家連接。

建立遊戲工作階段和玩家工作階段之後，與遊戲工作階段建立直接連接。

在 Local 命令提示字元視窗中，記錄訊息應顯示遊戲伺服器已傳送 `AcceptPlayerSession()` 請求，以驗證新玩家連接。如果您使用 AWS CLI 呼叫 `DescribePlayerSessions()`，玩家工作階段狀態應該會從預留變更為作用中。

6. 驗證遊戲伺服器是否正在向 Amazon GameLift Servers 服務回報遊戲與玩家狀態。

為了讓 Amazon GameLift Servers 能管理玩家需求並確實報告指標，遊戲伺服器必須向 Amazon GameLift Servers 回報各種狀態。驗證 Local 是否正在記錄與下列動作相關的事件。您也可以使用 AWS CLI 來追蹤狀態變更。

- 玩家中斷與遊戲工作階段的連線 – Amazon GameLift Servers 本機日誌訊息應會顯示您的遊戲伺服器呼叫 `RemovePlayerSession()`。此外，系統對 `DescribePlayerSessions()` 所進行的 AWS CLI 呼叫，應該會反映出狀態從 `Active` 變更為 `Completed`。您也可以呼叫 `DescribeGameSessions()`，檢查遊戲工作階段的目前玩家數目是否減一。
- 遊戲工作階段結束 – Amazon GameLift Servers 本機日誌訊息應顯示您的遊戲伺服器呼叫 `TerminateGameSession()`。

Note

先前的指引是在結束遊戲工作階段 `TerminateGameSession()` 時呼叫。此方法已使用 Amazon GameLift Servers Server SDK v4.0.1 取代。請參閱 [結束遊戲工作階段](#)。

- 伺服器程序已終止 – Amazon GameLift Servers 本機日誌訊息應會顯示您的遊戲伺服器呼叫 `ProcessEnding()`。對的 AWS CLI 呼叫 `DescribeGameSessions()` 應反映狀態從 `Active` 變更為 `Terminated` (或 `Terminating`)。

本機 的變化

使用 Amazon GameLift Servers Local 時，請記住下列要項：

- 與 Amazon GameLift Servers Web 服務不同，Local 不會追蹤伺服器的運作狀態，亦不會啟動 `onProcessTerminate` 回呼。Local 只會停止記錄遊戲伺服器的執行狀況報告。
- 對於對 AWS SDK 的呼叫，機群 IDs 不會經過驗證，並且可以是符合參數要求 () 的任何字串值 `^fleet-\S+`。
- 使用 Local 建立的遊戲工作階段 ID 有不同的架構。它們包含字串 `local`，如下所示：

```
arn:aws:gamelift:local::gamesession/fleet-123/gsess-56961f8e-  
db9c-4173-97e7-270b82f0daa6
```

新增FlexMatch配對

使用 Amazon GameLift Servers FlexMatch 將玩家配對功能新增至您的 Amazon GameLift Servers 託管遊戲。您可以 FlexMatch 搭配自訂遊戲伺服器或 使用 Amazon GameLift Servers Realtime。

FlexMatch 會將配對服務與自訂規則引擎結合在一起。您根據對遊戲有意義的玩家屬性和遊戲模式，設計如何將玩家配對在一起。FlexMatch 管理評估正在尋找遊戲的玩家、與一或多個隊伍組成配對，以及啟動遊戲工作階段來託管配對的螺母和螺絲。

若要使用完整 FlexMatch 服務，您必須使用佇列設定您的託管資源。Amazon GameLift Servers 使用佇列來尋找跨多個區域和運算類型之遊戲的最佳可能託管位置。特別是，Amazon GameLift Servers 佇列可以在遊戲用戶端提供時，使用延遲資料來放置遊戲工作階段，讓玩家在玩遊戲時體驗到最低的延遲。

如需 FlexMatch 將配對整合到遊戲中的詳細說明的詳細資訊，請參閱下列 [Amazon GameLift Servers FlexMatch 開發人員指南](#) 主題：

- [如何Amazon GameLift ServersFlexMatch運作](#)
- [FlexMatch 整合步驟](#)

取得Amazon GameLift Servers執行個體的機群資料

在某些情況下，您的自訂遊戲組建或Amazon GameLift ServersRealtime指令碼可能需要機Amazon GameLift Servers群的相關資訊。例如，您的遊戲組建或指令碼可能包含程式碼以：

- 根據機群資料監控活動。
- 彙總指標以依機群資料追蹤活動。（許多遊戲會將此資料用於 LiveOps 活動。）
- 將相關資料提供給自訂遊戲服務，例如配對、額外容量擴展或測試。

在下列位置的每個執行個體上，機群資訊會以 JSON 檔案的形式提供：

- Windows：C:\GameMetadata\gamelift-metadata.json
- Linux：/local/gamemetadata/gamelift-metadata.json

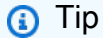
gamelift-metadata.json 檔案包含[Amazon GameLift Servers機群資源的屬性](#)。

範例 JSON 檔案：

```
{
  "buildArn": "arn:aws:gamelift:us-west-2:123456789012:build/build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "buildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "fleetArn": "arn:aws:gamelift:us-west-2:123456789012:fleet/fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "fleetDescription": "Test fleet for Really Fun Game v0.8",
  "fleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "name": "ReallyFunGameTestFleet08",
  "fleetType": "ON_DEMAND",
  "instanceRoleArn": "arn:aws:iam::123456789012:role/S3AccessForGameLift",
  "instanceType": "c5.large",
  "serverLaunchPath": "/local/game/reallyfungame.exe"
}
```

將遊戲與 整合 Amazon GameLift ServersRealtime

本主題提供Amazon GameLift Servers使用 Amazon GameLift ServersRealtime解決方案受管 的概觀。概觀說明此解決方案是否適合您的遊戲，以及 如何Amazon GameLift ServersRealtime支援多玩家遊戲。



Tip

[立即開始使用 Amazon GameLift Servers](#)

什麼是Realtime伺服器？

Realtime 伺服器是輕量且ready-to-go遊戲伺服器，Amazon GameLift Servers可讓您與多玩家遊戲搭配使用。Realtime 伺服器會移除自訂遊戲伺服器的開發、測試和部署程序。此解決方案有助於將完成遊戲所需的時間和精力降至最低。

主要功能

- 遊戲用戶端和伺服器互動的完整網路堆疊
- 核心遊戲伺服器功能
- 可自訂的伺服器邏輯
- Realtime 組態和伺服器邏輯的即時更新
- FlexMatch 配對
- 彈性控制託管資源

透過建立機群並提供組態指令碼來設定Realtime伺服器。

如何Amazon GameLift ServersRealtime管理遊戲工作階段

您可以將遊戲工作階段管理的自訂邏輯建置為Realtime指令碼。您可以編寫程式碼來存取伺服器特定的物件、使用回呼新增事件驅動邏輯，或根據非事件案例新增邏輯。

Realtime 用戶端和伺服器如何互動

在遊戲工作階段期間，遊戲用戶端會透過後端服務傳送訊息至Realtime伺服器來互動。然後，後端服務會在遊戲用戶端之間轉送訊息，以交換活動、遊戲狀態和相關遊戲資料。

此外，您可以將遊戲邏輯新增到 Realtime 指令碼，來自訂用戶端和伺服器的互動方式。透過自訂遊戲邏輯，Realtime 伺服器可能會實作回呼來啟動事件驅動的回應。

通訊協定

Realtime 伺服器和連線遊戲用戶端透過兩個通道進行通訊：提供可靠交付的 TCP 連線，以及提供快速交付的 UDP 通道。建立訊息時，遊戲用戶端會根據訊息的本質選擇要使用的通訊協定。根據預設，訊息交付會設為 UDP。如果無法使用 UDP 頻道，會使用 TCP 做為備用來 Amazon GameLift Servers 傳送訊息。

訊息內容

訊息內容包含兩個元素：必要的操作碼 (opCode) 和選用的承載。訊息的 opCode 可識別特定的玩家活動或遊戲事件，承載會提供與操作程式碼相關的其他資料。這兩個元素都是由開發人員定義。您的遊戲用戶端會根據收到的訊息中的 opCodes 來運作。

玩家群組

Amazon GameLift Servers Realtime 提供管理玩家群組的功能。根據預設，會將連線到遊戲的所有玩家 Amazon GameLift Servers 置於「所有玩家」群組中。此外，開發人員可以為其遊戲設定其他群組，且玩家可以同時是多個群組的成員。群組成員可以傳送訊息並與群組中的所有玩家共用遊戲資料。群組的一個可能用途是設定玩家團隊和管理團隊通訊。

Amazon GameLift Servers Realtime 使用 TLS 憑證

使用時 Amazon GameLift Servers Realtime，伺服器身分驗證和資料封包加密會內建在服務中。當您開啟 TLS 憑證產生時，會啟用這些安全功能。當遊戲用戶端嘗試與 Realtime 伺服器連線時，伺服器會自動回應用戶端驗證的 TLS 憑證。會使用 TLS for TCP (WebSockets) 通訊和 DTLS for UDP 流量 Amazon GameLift Servers 來處理加密。

自訂 Realtime 伺服器

Realtime 伺服器做為無狀態轉送伺服器執行。Realtime 伺服器會在連接到遊戲的遊戲用戶端之間轉送訊息和遊戲資料的封包。不過，Realtime 伺服器不會評估訊息、處理資料或執行任何遊戲邏輯。以這種方式使用，每個遊戲用戶端都會維護自己的遊戲狀態檢視，並透過轉送伺服器提供更新給其他玩家。每個遊戲用戶端負責整合這些更新和調節自己的遊戲狀態。

您可以透過將新增至 Realtime 指令碼功能來自訂您的伺服器。例如，透過遊戲邏輯，您可以使用遊戲狀態的伺服器授權檢視來建置具狀態的遊戲。

Amazon GameLift Servers 定義 Realtime 指令碼的一組伺服器端回呼。實作這些回呼，以將事件驅動的功能新增到您的伺服器。例如，您可以：

- 當遊戲用戶端嘗試連接到伺服器時對玩家進行身分驗證。
- 驗證玩家是否可以根據請求加入群組。
- 決定何時從特定玩家或目標玩家傳送訊息，或執行額外的處理以回應。
- 當玩家離開群組或中斷與伺服器的連線時，通知所有玩家。
- 檢視遊戲工作階段物件或訊息物件的內容，並使用資料。

部署和更新 Amazon GameLift Servers Realtime

的主要優點 Amazon GameLift Servers Realtime 是能夠隨時更新指令碼。當您更新指令碼時，會在幾分鐘內將新版本 Amazon GameLift Servers 分發至所有託管資源。Amazon GameLift Servers 部署新的指令碼之後，在該時間點之後建立的所有新遊戲工作階段都會使用新的指令碼版本。（現有的遊戲工作階段將繼續使用原始版本。）

開始將遊戲與 整合 Amazon GameLift Servers Realtime：

- [整合 的遊戲用戶端 Amazon GameLift Servers Realtime](#)
- [建立 Realtime 指令碼](#)

整合 的遊戲用戶端 Amazon GameLift Servers Realtime

此主題說明如何準備您的遊戲用戶端，以便能夠加入和參與 Amazon GameLift Servers 託管的遊戲工作階段。

要準備您的遊戲用戶端需要兩組任務：

- 設定您的遊戲用戶端，以取得有關現有遊戲的資訊、請求配對、啟動新的遊戲工作階段，以及為玩家預留遊戲工作階段位置。
- 啟用您的遊戲用戶端，以加入在 Realtime 伺服器上託管的遊戲工作階段並交換訊息。

尋找或建立遊戲工作階段和玩家工作階段

設定遊戲用戶端，以尋找或啟動遊戲工作階段、請求 FlexMatch 配對，以及透過建立玩家工作階段，來為遊戲中的玩家預留空間。最佳實務是建立後端服務，並在遊戲用戶端動作觸發時，使用它直接請求 Amazon GameLift Servers 服務。然後，後端服務會將相關回應轉送回遊戲用戶端。

1. 將 AWS SDK 新增至您的遊戲用戶端、初始化 Amazon GameLift Servers 用戶端，並將其設定為使用機群和佇列中的託管資源。AWS 開發套件提供多種語言；請參閱 Amazon GameLift Servers SDKs [對於遊戲用戶端服務](#)。
2. 將 Amazon GameLift Servers 功能新增至您的後端服務。如需更詳細的說明，請參閱 [Amazon GameLift Servers 新增至您的遊戲用戶端](#) 和 [新增 FlexMatch 配對](#)。最佳實務是使用遊戲工作階段置放來建立新的遊戲工作階段。此方法可讓您充分利用快速且智慧地放置新遊戲工作階段 Amazon GameLift Servers 的能力，以及使用玩家延遲資料將遊戲延遲降至最低。您的後端服務至少必須能夠請求新的遊戲工作階段，並處理遊戲工作階段資料以回應。您可能也會想要新增功能，來搜尋和取得現有遊戲工作階段的資訊，並請求玩家工作階段，有效地在現有的遊戲工作階段中預留玩家位置。
3. 將連線資訊傳回遊戲用戶端。後端服務會接收遊戲工作階段和玩家工作階段物件，以回應對 Amazon GameLift Servers 服務的請求。這些物件包含資訊，特別是遊戲用戶端連線到在 Realtime 伺服器上所執行遊戲工作階段所需的連線詳細資訊 (IP 地址和連接埠) 以及玩家工作階段 ID。

連線至 上的遊戲 Amazon GameLift Servers Realtime

讓您的遊戲用戶端與 Realtime 伺服器上託管的遊戲工作階段直接連接，並與伺服器和其他玩家交換訊息。

1. 取得適用於 的用戶端 SDK Amazon GameLift Servers Realtime、建置它，並將其新增至您的遊戲用戶端專案。如需 SDK 需求以及如何建置用戶端程式庫的詳細資訊，請參閱 README 檔案。
2. 使用指定用戶端/伺服器連線類型的用戶端組態來呼叫 [Client\(\)](#)。

Note

如果您連線到在使用 TLS 憑證之安全機群上執行的 Realtime 伺服器，您必須指定安全連線類型。

3. 將以下功能新增到您的遊戲用戶端。如需詳細資訊，請參閱 [Amazon GameLift Servers Realtime 用戶端 API \(C#\) 參考](#)。
 - 連線及從遊戲中斷連線
 - [Connect\(\)](#)
 - [Disconnect\(\)](#)
 - 將訊息傳送給目標收件人
 - [SendMessage\(\)](#)

- 接收和處理訊息
 - [OnDataReceived\(\)](#)
 - 加入群組和離開玩家群組
 - [JoinGroup\(\)](#)
 - [RequestGroupMembership\(\)](#)
 - [LeaveGroup\(\)](#)
4. 視需要為用戶端回呼設定事件處理器。請參閱 [Amazon GameLift ServersRealtime 用戶端 API \(C#\)](#) [參考：非同步回呼](#)。

使用已啟用 TLS 憑證產生的 Realtime 機群時，系統會自動使用 TLS 憑證來驗證伺服器。TCP 和 UDP 流量會在傳送中加密，以提供傳輸層安全性。TCP 流量會使用 TLS 1.2 加密，而 UDP 流量則使用 DTLS 1.2 加密。

遊戲用戶端範例

基本即時用戶端 (C#)

此範例說明 與用戶端 SDK (C#) 的基本遊戲用戶端整合 Amazon GameLift ServersRealtime。如下所示，範例會初始化 Realtime 用戶端物件、設定事件處理常式並實作用戶端回呼、連接到 Realtime 伺服器、傳送訊息，以及中斷連線。

```
using System;
using System.Text;
using Aws.GameLift.Realtime;
using Aws.GameLift.Realtime.Event;
using Aws.GameLift.Realtime.Types;

namespace Example
{
    /**
     * An example client that wraps the client SDK for Amazon GameLift Servers Realtime
     *
     * You can redirect logging from the SDK by setting up the LogHandler as such:
     * ClientLogger.LogHandler = (x) => Console.WriteLine(x);
     *
     */
    class RealTimeClient
    {
        public Aws.GameLift.Realtime.Client Client { get; private set; }
    }
}
```

```

        // An opcode defined by client and your server script that represents a custom
message type
        private const int MY_TEST_OP_CODE = 10;

        /// Initialize a client for Amazon GameLift Servers Realtime and connect to a
player session.
        /// <param name="endpoint">The DNS name that is assigned to Realtime server</
param>
        /// <param name="remoteTcpPort">A TCP port for the Realtime server</param>
        /// <param name="listeningUdpPort">A local port for listening to UDP traffic</
param>
        /// <param name="connectionType">Type of connection to establish between client
and the Realtime server</param>
        /// <param name="playerSessionId">The player session ID that is assigned to the
game client for a game session </param>
        /// <param name="connectionPayload">Developer-defined data to be used during
client connection, such as for player authentication</param>
        public RealTimeClient(string endpoint, int remoteTcpPort, int listeningUdpPort,
ConnectionType connectionType,
                                string playerSessionId, byte[] connectionPayload)
        {
            // Create a client configuration to specify a secure or unsecure connection
type
            // Best practice is to set up a secure connection using the connection type
RT_OVER_WSS_DTLS_TLS12.
            ClientConfiguration clientConfiguration = new ClientConfiguration()
            {
                // C# notation to set the field ConnectionType in the new instance of
ClientConfiguration
                ConnectionType = connectionType
            };

            // Create a Realtime client with the client configuration
            Client = new Client(clientConfiguration);

            // Initialize event handlers for the Realtime client
            Client.ConnectionOpen += OnOpenEvent;
            Client.ConnectionClose += OnCloseEvent;
            Client.GroupMembershipUpdated += OnGroupMembershipUpdate;
            Client.DataReceived += OnDataReceived;

            // Create a connection token to authenticate the client with the Realtime
server

```

```

        // Player session IDs can be retrieved using AWS SDK for Amazon GameLift
Servers
        ConnectionToken connectionToken = new ConnectionToken(playerSessionId,
connectionPayload);

        // Initiate a connection with the Realtime server with the given connection
information
        Client.Connect(endpoint, remoteTcpPort, listeningUdpPort, connectionToken);
    }

    public void Disconnect()
    {
        if (Client.Connected)
        {
            Client.Disconnect();
        }
    }

    public bool IsConnected()
    {
        return Client.Connected;
    }

    /// <summary>
    /// Example of sending to a custom message to the server.
    ///
    /// Server could be replaced by known peer Id etc.
    /// </summary>
    /// <param name="intent">Choice of delivery intent i.e. Reliable, Fast etc. </
param>
    /// <param name="payload">Custom payload to send with message</param>
    public void SendMessage(DeliveryIntent intent, string payload)
    {
        Client.SendMessage(Client.NewMessage(MY_TEST_OP_CODE)
            .WithDeliveryIntent(intent)
            .WithTargetPlayer(Constants.PLAYER_ID_SERVER)
            .WithPayload(StringToBytes(payload)));
    }

    /**
     * Handle connection open events
     */
    public void OnOpenEvent(object sender, EventArgs e)
    {

```

```
}

/**
 * Handle connection close events
 */
public void OnCloseEvent(object sender, EventArgs e)
{
}

/**
 * Handle Group membership update events
 */
public void OnGroupMembershipUpdate(object sender, GroupMembershipEventArgs e)
{
}

/**
 * Handle data received from the Realtime server
 */
public virtual void OnDataReceived(object sender, DataReceivedEventArgs e)
{
    switch (e.OpCode)
    {
        // handle message based on OpCode
        default:
            break;
    }
}

/**
 * Helper method to simplify task of sending/receiving payloads.
 */
public static byte[] StringToBytes(string str)
{
    return Encoding.UTF8.GetBytes(str);
}

/**
 * Helper method to simplify task of sending/receiving payloads.
 */
public static string BytesToString(byte[] bytes)
{
    return Encoding.UTF8.GetString(bytes);
}
```



```
}  
}
```

建立Realtime指令碼

若要將 Amazon GameLift ServersRealtime 用於您的遊戲，您需要提供指令碼（以某些 JavaScript 程式碼的形式）來設定和選擇性地自訂的機群Amazon GameLift ServersRealtime。本主題涵蓋建立 Realtime 指令碼的重要步驟。指令碼準備就緒後，請上傳到 Amazon GameLift Servers 服務，並使用它來建立叢集（請參閱[部署的指令碼 Amazon GameLift ServersRealtime](#)）。

若要準備指令碼以搭配使用Amazon GameLift ServersRealtime，請將下列功能新增至您的Realtime指令碼。

管理遊戲工作階段生命週期（必要）

Realtime 指令碼必須至少包含 `Init()` 函數，用來準備 Realtime 伺服器啟動遊戲工作階段。我們也強烈建議您提供終止遊戲工作階段的方式，確保新的遊戲工作階段可以繼續在您的叢集上啟動。

`Init()` 回呼函數在受到呼叫的時候會獲得傳遞一個 Realtime 工作階段物件，其中包含 Realtime 伺服器的介面。請參閱 [Amazon GameLift ServersRealtime 介面](#) 以取得此介面的詳細資訊。

若要順利結束遊戲工作階段，指令碼也必須呼叫 Realtime 伺服器的 `session.processEnding` 函數。這需要一些機制，來判斷結束工作階段的時機。指令碼範例程式碼會示範簡易的機制，檢查玩家連線，並在指定的時間長度內沒有任何玩家連線到工作階段時觸發遊戲工作階段的終止。

Amazon GameLift ServersRealtime 具有最基本的組態 - 伺服器程序初始化和終止 - 基本上充當無狀態轉送伺服器。Realtime 伺服器會在連線到遊戲的遊戲用戶端之間轉送訊息和遊戲資料，但不會採取獨立的動作來處理資料或執行邏輯。根據您遊戲的需求，您可以選用地新增遊戲邏輯，由遊戲事件或其他機制觸發。

新增伺服器端遊戲邏輯（選用）

您可以選用地將遊戲邏輯新增到您的 Realtime 指令碼。例如，您可以執行以下任何或所有的作業：指令碼範例程式碼會提供說明。請參閱 [的指令碼參考 Amazon GameLift ServersRealtime](#)。

- 新增事件驅動邏輯。實作回呼函數，以回應用戶端 – 伺服器事件。如需完整的回呼清單，請參閱 [的指令碼回呼 Amazon GameLift ServersRealtime](#)。
- 透過傳送訊息至伺服器來觸發邏輯。針對從遊戲用戶端傳送到伺服器的訊息，建立一組特殊操作程式碼，並新增函數來處理接收。使用回呼 `onMessage`，並使用 `gameMessage` 介面來剖析訊息內容（請參閱 [gameMessage.opcode](#)）。

- 啟用遊戲邏輯以存取您的其他 AWS 資源。如需詳細資訊，請參閱[與機群中的其他 AWS 資源通訊](#)。
- 允許遊戲邏輯存取其執行所在執行個體的機群資訊。如需詳細資訊，請參閱[取得Amazon GameLift Servers執行個體的機群資料](#)。

Amazon GameLift ServersRealtime 指令碼範例

此範例說明部署所需的基本指令碼，Amazon GameLift ServersRealtime以及一些自訂邏輯。它包含必要的 `Init()` 函數，並會使用計時器機制來根據沒有任何玩家連線的時間長度，觸發遊戲工作階段的終止。它還包含一些自訂邏輯的勾點，其中包括一些回呼實作。

```
// Example Realtime Server Script
'use strict';

// Example override configuration
const configuration = {
  pingIntervalTime: 30000,
  maxPlayers: 32
};

// Timing mechanism used to trigger end of game session. Defines how long, in
// milliseconds, between each tick in the example tick loop
const tickTime = 1000;

// Defines how to long to wait in Seconds before beginning early termination check in
// the example tick loop
const minimumElapsedTime = 120;

var session; // The Realtime server session object
var logger; // Log at appropriate level
  via .info(), .warn(), .error(), .debug()
var startTime; // Records the time the process started
var activePlayers = 0; // Records the number of connected players
var onProcessStartedCalled = false; // Record if onProcessStarted has been called

// Example custom op codes for user-defined messages
// Any positive op code number can be defined here. These should match your client
// code.
const OP_CODE_CUSTOM_OP1 = 111;
const OP_CODE_CUSTOM_OP1_REPLY = 112;
const OP_CODE_PLAYER_ACCEPTED = 113;
const OP_CODE_DISCONNECT_NOTIFICATION = 114;
```

```
// Example groups for user-defined groups
// Any positive group number can be defined here. These should match your client code.
// When referring to user-defined groups, "-1" represents all groups, "0" is reserved.
const RED_TEAM_GROUP = 1;
const BLUE_TEAM_GROUP = 2;

// Called when game server is initialized, passed server's object of current session
function init(rtSession) {
    session = rtSession;
    logger = session.getLogger();
}

// On Process Started is called when the process has begun and we need to perform any
// bootstrapping. This is where the developer should insert any code to prepare
// the process to be able to host a game session, for example load some settings or set
// state
//
// Return true if the process has been appropriately prepared and it is okay to invoke
// the
// GameLift ProcessReady() call.
function onProcessStarted(args) {
    onProcessStartedCalled = true;
    logger.info("Starting process with args: " + args);
    logger.info("Ready to host games...");

    return true;
}

// Called when a new game session is started on the process
function onStartGameSession(gameSession) {
    // Complete any game session set-up

    // Set up an example tick loop to perform server initiated actions
    startTime = getTimeInS();
    tickLoop();
}

// Handle process termination if the process is being terminated by Amazon GameLift
// Servers
// You do not need to call ProcessEnding here
function onProcessTerminate() {
    // Perform any clean up
}
```

```
// Return true if the process is healthy
function onHealthCheck() {
    return true;
}

// On Player Connect is called when a player has passed initial validation
// Return true if player should connect, false to reject
function onPlayerConnect(connectMsg) {
    // Perform any validation needed for connectMsg.payload, connectMsg.peerId
    return true;
}

// Called when a Player is accepted into the game
function onPlayerAccepted(player) {
    // This player was accepted -- let's send them a message
    const msg = session.newTextGameMessage(OP_CODE_PLAYER_ACCEPTED, player.peerId,
                                            "Peer " + player.peerId + " accepted");
    session.sendReliableMessage(msg, player.peerId);
    activePlayers++;
}

// On Player Disconnect is called when a player has left or been forcibly terminated
// Is only called for players that actually connected to the server and not those
// rejected by validation
// This is called before the player is removed from the player list
function onPlayerDisconnect(peerId) {
    // send a message to each remaining player letting them know about the disconnect
    const outMessage = session.newTextGameMessage(OP_CODE_DISCONNECT_NOTIFICATION,
                                                  session.getServerId(),
                                                  "Peer " + peerId + " disconnected");
    session.getPlayers().forEach((player, playerId) => {
        if (playerId !== peerId) {
            session.sendReliableMessage(outMessage, playerId);
        }
    });
    activePlayers--;
}

// Handle a message to the server
function onMessage(gameMessage) {
    switch (gameMessage.opCode) {
        case OP_CODE_CUSTOM_OP1: {
            // do operation 1 with gameMessage.payload for example sendToGroup
        }
    }
}
```

```

        const outMessage = session.newTextGameMessage(OP_CODE_CUSTOM_OP1_REPLY,
session.getServerId(), gameMessage.payload);
        session.sendGroupMessage(outMessage, RED_TEAM_GROUP);
        break;
    }
}
}

// Return true if the send should be allowed
function onSendToPlayer(gameMessage) {
    // This example rejects any payloads containing "Reject"
    return (!gameMessage.getPayloadAsText().includes("Reject"));
}

// Return true if the send to group should be allowed
// Use gameMessage.getPayloadAsText() to get the message contents
function onSendToGroup(gameMessage) {
    return true;
}

// Return true if the player is allowed to join the group
function onPlayerJoinGroup(groupId, peerId) {
    return true;
}

// Return true if the player is allowed to leave the group
function onPlayerLeaveGroup(groupId, peerId) {
    return true;
}

// A simple tick loop example
// Checks to see if a minimum amount of time has passed before seeing if the game has
// ended
async function tickLoop() {
    const elapsedTime = getTimeInS() - startTime;
    logger.info("Tick... " + elapsedTime + " activePlayers: " + activePlayers);

    // In Tick loop - see if all players have left early after a minimum period of time
    // has passed
    // Call processEnding() to terminate the process and quit
    if ( (activePlayers == 0) && (elapsedTime > minimumElapsedTime)) {
        logger.info("All players disconnected. Ending game");
        const outcome = await session.processEnding();
        logger.info("Completed process ending with: " + outcome);
    }
}

```

```
        process.exit(0);
    }
    else {
        setTimeout(tickLoop, tickTime);
    }
}

// Calculates the current time in seconds
function getTimeInS() {
    return Math.round(new Date().getTime()/1000);
}

exports.ssExports = {
    configuration: configuration,
    init: init,
    onProcessStarted: onProcessStarted,
    onMessage: onMessage,
    onPlayerConnect: onPlayerConnect,
    onPlayerAccepted: onPlayerAccepted,
    onPlayerDisconnect: onPlayerDisconnect,
    onSendToPlayer: onSendToPlayer,
    onSendToGroup: onSendToGroup,
    onPlayerJoinGroup: onPlayerJoinGroup,
    onPlayerLeaveGroup: onPlayerLeaveGroup,
    onStartGameSession: onStartGameSession,
    onProcessTerminate: onProcessTerminate,
    onHealthCheck: onHealthCheck
};
```

部署遊戲伺服器軟體以Amazon GameLift Servers託管

在託管資源上安裝多玩家遊戲伺服器軟體、啟動遊戲伺服器程序，並準備好為玩家託管遊戲，以部署多玩家遊戲伺服器軟體。讓遊戲伺服器軟體準備好進行部署的步驟取決於您使用Amazon GameLift Servers的解決方案類型。在所有案例中，部署的遊戲伺服器軟體會與服務互動Amazon GameLift Servers，以處理遊戲工作階段放置並傳達遊戲用戶端的連線詳細資訊。

您的遊戲伺服器軟體必須先與 整合Amazon GameLift Servers，如中所述[準備的遊戲 Amazon GameLift Servers](#)。

本節中的主題提供如何讓您的軟體準備好部署至下列案例的指引。

- 自訂遊戲伺服器軟體
 - 適用於Amazon GameLift Servers受管 EC2 託管
 - Amazon GameLift Servers 受管容器託管
 - 對於 Amazon GameLift Servers Anywhere 託管
- Amazon GameLift ServersRealtime Amazon GameLift Servers受管 EC2 託管的自訂指令碼

主題

- [部署用於Amazon GameLift Servers託管的自訂伺服器建置](#)
- [建置的容器映像 Amazon GameLift Servers](#)
- [部署的指令碼 Amazon GameLift ServersRealtime](#)

部署用於Amazon GameLift Servers託管的自訂伺服器建置

將遊戲伺服器與 整合後 Amazon GameLift Servers (請參閱 [準備的遊戲 Amazon GameLift Servers](#))，請將遊戲伺服器軟體安裝到運算資源上以進行託管。此程序會根據您使用的Amazon GameLift Servers託管類型而有所不同。

部署受管託管

如果您使用的是Amazon GameLift Servers受管 EC2 託管，則必須封裝遊戲伺服器軟體並將其上傳至Amazon GameLift Servers。當您建立受管機群時，Amazon GameLift Servers會自動將其部署到每個機群執行個體。

本節中的主題說明如何封裝您的建置檔案以進行上傳、建立選用的建置安裝指令碼，然後使用 [AWS Command Line Interface \(AWS CLI\)](#) 或 AWS SDK 上傳檔案。

部署 for Anywhere 託管

如果您使用 Amazon GameLift Servers Anywhere 機群進行自我管理託管，您有責任將遊戲伺服器軟體安裝到機群中的每個運算，並保持更新狀態。

當整合的遊戲伺服器程序開始執行時，它會自動初始化並與 Amazon GameLift Servers 服務建立通訊。伺服器程序會從提示啟動遊戲工作階段，Amazon GameLift Servers 並將活動回報給服務。

主題

- [封裝遊戲建置檔案](#)
- [新增建置安裝指令碼](#)
- [建立受管託管的 Amazon GameLift Servers 建置資源](#)
- [更新 Amazon GameLift Servers 受管託管的遊戲伺服器建置](#)

封裝遊戲建置檔案

將已設定的遊戲伺服器上傳至之前 Amazon GameLift Servers，請將遊戲建置檔案封裝至建置目錄。使用 EC2 受管機群託管時需要此程序，使用 Anywhere 機群託管時最佳實務。建置目錄應包含執行遊戲伺服器和主機遊戲工作階段所需的所有元件。這可能包括下列項目：

- 遊戲伺服器二進位檔 – 執行遊戲伺服器所需的二進位檔案。組建可以包含多個遊戲伺服器的二進位檔，這些遊戲伺服器是建置在相同平台上執行。如需支援的平台清單，請參閱 [取得 Amazon GameLift Servers 開發工具](#)。
- 相依性 – 遊戲伺服器可執行檔執行時所需的任何相依檔案。範例包括資產、設定檔和相依程式庫。

Note

對於使用適用於 C++ 的伺服器 SDK 建立 Amazon GameLift Servers 的遊戲組建（包括使用 Unreal 外掛程式建立的遊戲組建），請包含與您建置伺服器 SDK 的相同 OpenSSL 版本的 OpenSSL DLL。如需詳細資訊，請參閱伺服器 SDK README 檔案。

- 安裝指令碼（選用）– 指令碼檔案，用於處理在 Amazon GameLift Servers 託管伺服器上安裝遊戲組建的任務。將此檔案放在建置目錄的根目錄。會在建立機群時 Amazon GameLift Servers 執行安裝指令碼。

您可以在您的建置中設定任何應用程式，包括您的安裝指令碼，以在其他 AWS 服務上安全地存取您的資源。如需如何執行此操作的資訊，請參閱 [與機群中的其他 AWS 資源通訊](#)。

封裝建置檔案之後，請確定您的遊戲伺服器可以在目標作業系統的乾淨安裝上執行，以確認包含所有必要的相依性，且您的安裝指令碼正確無誤。

新增建置安裝指令碼

為遊戲建置的作業系統 (OS) 建立安裝指令碼：

- Windows：建立名為 `install.bat` 的批次檔案。
- Linux：建立名為 `install.sh` 的 shell 指令碼檔案。

在建立安裝指令碼時，請注意下列資訊：

- 指令碼無法接受任何使用者輸入。
- Amazon GameLift Servers 在下列位置的託管伺服器上，安裝組建並重新建立組建套件中的檔案目錄：
 - Windows 機群：C:\game
 - Linux 機群：/local/game
- 在 Linux 機群的安裝程序期間，執行身分使用者對執行個體檔案結構的存取有限。此使用者擁有安裝建置檔案之目錄的完整權限。如果您的安裝指令碼執行需要管理員許可的動作，請使用指定管理員存取權 `sudo`。Windows 機群的執行身分使用者預設具有管理員許可。與安裝指令碼相關的許可失敗會產生事件訊息，指出指令碼有問題。
- 在 Linux 上，Amazon GameLift Servers 支援常見的 shell 解譯器語言，例如 `bash`。在安裝指令碼的開頭新增 shebang (例如 `#!/bin/bash`)。若要驗證您偏好的 shell 命令是否支援，請遠端存取作用中的 Linux 執行個體並開啟 shell 提示。如需詳細資訊，請參閱 [遠端連線至 Amazon GameLift Servers 機群執行個體](#)。
- 安裝指令碼無法依賴 VPC 對等連線。在將建置 Amazon GameLift Servers 安裝到機群執行個體之前，無法使用 VPC 互連連線。

Example Windows 安裝 bash 檔案

此範例 `install.bat` 檔案會安裝遊戲伺服器所需的 Visual C++ 執行期元件，並將結果寫入日誌檔案。指令碼在根目錄的建置套件中包含元件檔案。

```
vcredis_x64.exe /install /quiet /norestart /log c:\game\vcredis_2013_x64.log
```

Example Linux 安裝 shell 指令碼

此範例install.sh檔案在安裝指令碼中使用 bash，並將結果寫入日誌檔案。

```
#!/bin/bash
echo 'Hello World' > install.log
```

此範例install.sh檔案示範如何使用 Amazon CloudWatch 代理程式來收集系統層級和自訂指標，以及處理日誌輪換。由於是在服務 VPC 中Amazon GameLift Servers執行，因此您必須授予代表您擔任 AWS Identity and Access Management (IAM) 角色的Amazon GameLift Servers許可。若要允許 Amazon GameLift Servers 擔任角色，請建立包含 AWS 受管政策 的角色CloudWatchAgentAdminPolicy，並在建立機群時使用該角色。

```
sudo yum install -y amazon-cloudwatch-agent
sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
sudo yum install -y collectd
cat <<'EOF' > /tmp/config.json
{
    "agent": {
        "metrics_collection_interval": 60,
        "run_as_user": "root",
        "credentials": {
            "role_arn": "arn:aws:iam::account#:role/rolename"
        }
    },
    "logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "/tmp/log",
                        "log_group_name": "gllog",
                        "log_stream_name": "{instance_id}"
                    }
                ]
            }
        }
    },
    "metrics": {
```

```
"namespace": "GL_Metric",
  "append_dimensions": {
    "ImageId": "${aws:ImageId}",
    "InstanceId": "${aws:InstanceId}",
    "InstanceType": "${aws:InstanceType}"
  },
  "metrics_collected": {
    // Configure metrics you want to collect.
    // For more information, see Manually create or edit the CloudWatch agent configuration file.
  }
}
EOF
sudo mv /tmp/config.json /opt/aws/amazon-cloudwatch-agent/bin/config.json
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
sudo systemctl enable amazon-cloudwatch-agent.service
```

建立受管託管的Amazon GameLift Servers建置資源

在建立建置並上傳檔案時，您有幾個選項：

- [從檔案目錄建立組建](#)。這是最簡單且最常用的選項。
- [使用 Amazon Simple Storage Service \(Amazon S3\) 中的檔案建立組建](#)。使用此選項，您可以在 Amazon S3 中管理建置版本。

使用這兩種方法，Amazon GameLift Servers 會使用唯一的建置 ID 和其他中繼資料建立新的建置資源。組建以初始化狀態開始。在 Amazon GameLift Servers 取得遊戲伺服器檔案後，組建會移至就緒狀態。

當組建準備就緒時，您可以將其部署到新的 Amazon GameLift Servers 機群。如需詳細資訊，請參閱 [建立 Amazon GameLift Servers 受管 EC2 機群](#)。Amazon GameLift Servers 當設定新機群時，它會將建置檔案下載至每個機群執行個體，並安裝建置檔案。

從檔案目錄建立組建

若要建立存放在任何位置的遊戲組建，包括本機目錄，請使用 [upload-build](#) AWS CLI 命令。此命令會在 Amazon GameLift Servers 中建立新的建置記錄，並從您指定的位置上傳檔案。

傳送上傳請求。在命令列視窗中，輸入下列 `upload-build` 命令和參數。

```
aws gamelift upload-build \  
  --name user-defined name of build \  
  --operating-system supported OS \  
  --server-sdk-version server SDK for Amazon GameLift Servers version \  
  --build-root build path \  
  --build-version user-defined build number \  
  --region region name
```

- operating-system – 遊戲伺服器的建置執行時間環境。您必須指定作業系統值。您稍後無法更新此項目。
- server-sdk-version – 遊戲Amazon GameLift Servers伺服器整合的伺服器 SDK 版本。如果您未提供值，Amazon GameLift Servers會使用預設值 4.0.2。如果您指定的伺服器 SDK 版本不正確，則呼叫 InitSdk以建立Amazon GameLift Servers與服務的連線時，遊戲伺服器建置可能會失敗。
- build-root – 建置檔案的目錄路徑。
- name – 新建置的描述性名稱。
- build-version – 組建檔案的版本詳細資訊。
- region – 您要建立組建 AWS 的區域。在您計劃部署機群的區域中建立建置。如果您要在多個區域中部署遊戲，請在每個區域中建立組建。

Note

使用 檢視您目前的預設區域[aws configure get region](#)。若要變更您的預設區域，請使用 [aws configure set region *region name*](#)命令。

範例

```
aws gamelift upload-build \  
  --operating-system AMAZON_LINUX_2023 \  
  
  --server-sdk-version "5.0.0" \  
  --build-root "~/mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-west-2
```

```
aws gamelift upload-build \  
  --operating-system WINDOWS_2016 \
```

```
--server-sdk-version "5.0.0" \  
--build-root "C:\mygame" \  
--name "My Game Nightly Build" \  
--build-version "build 255" \  
--region us-west-2
```

為了回應您的上傳請求，Amazon GameLift Servers提供上傳進度。成功上傳時，會Amazon GameLift Servers傳回新的建置記錄 ID。上傳時間取決於遊戲檔案的大小和連線速度。

使用 Amazon S3 中的檔案建立組建

您可以將建置檔案存放在 Amazon S3 中，並從Amazon GameLift Servers該處上傳到。建立建置時，您可以指定 S3 儲存貯體位置，並直接從 Amazon S3 Amazon GameLift Servers擷取建置檔案。

建立建置資源

1. 將您的建置檔案儲存在 Amazon S3 中。建立包含封裝建置檔案的 .zip 檔案，並將其上傳至 中的 S3 儲存貯體 AWS 帳戶。請記下儲存貯體標籤和檔案名稱，在建立Amazon GameLift Servers組建時將需要這些標籤。
2. 讓 Amazon GameLift Servers 可以存取您的建置檔案。遵循 中的指示建立 IAM 角色[在 Amazon S3 中存取遊戲組建檔案](#)。建立角色後，請記下新角色的 Amazon Resource Name (ARN)，建立組建時將需要此名稱。
3. 建立組建。使用 Amazon GameLift Servers主控台或 AWS CLI 建立新的建置記錄。您必須擁有 PassRole許可，如 中所述的[IAM 許可範例 Amazon GameLift Servers](#)。

Console

1. 在[Amazon GameLift Servers主控台](#)的導覽窗格中，選擇託管、建置。
2. 在建置頁面上，選擇建立建置。
3. 在建立建置頁面的建置設定下，執行下列動作：
 - a. 在名稱中，輸入指令碼名稱。
 - b. 在版本中，輸入版本。由於您可以更新組建的內容，版本資料可協助您追蹤更新。
 - c. 針對作業系統 (OS)，選擇遊戲伺服器建置的作業系統。您稍後無法更新此值。
 - d. 針對遊戲伺服器組建，輸入您上傳到 Amazon S3 的組建物件的 S3 URI，然後選擇物件版本。Amazon S3 如果您不記得 Amazon S3 URI 和物件版本，請選擇瀏覽 S3 並搜尋建置物件。

- e. 針對 IAM 角色，選擇您建立的角色，以授予對 S3 儲存貯體和建置物件的 Amazon GameLift Servers 存取權。
4. (選用) 在標籤下，輸入索引鍵和值對，將標籤新增至組建。
5. 選擇 Create (建立)。

Amazon GameLift Servers 會將 ID 指派給新組建，並上傳指定的 .zip 檔案。您可以在組建頁面上檢視新組建，包括狀態。

AWS CLI

若要定義新建置並上傳伺服器建置檔案，請使用 [create-build](#) 命令。

1. 開啟命令列視窗，並切換到您可以使用的目錄 AWS CLI。
2. 輸入下列 create-build 命令：

```
aws gamelift create-build \  
  --name user-defined name of build \  
  --server-sdk-version server SDK for Amazon GameLift Servers version \  
  --operating-system supported OS \  
  --build-version user-defined build number \  
  --storage-location "Bucket"=S3 bucket label, "Key"=Build .zip file name, "RoleArn"=Access role ARN} \  
  --region region name
```

- name – 新建置的描述性名稱。
- server-sdk-version – Amazon GameLift Servers 您用來整合遊戲伺服器與的伺服器 SDK 版本 Amazon GameLift Servers。如果您未提供值，Amazon GameLift Servers 會使用預設值 4.0.2。
- operating-system – 遊戲伺服器的建置執行期環境。您必須指定作業系統值。您稍後無法更新此項目。
- build-version – 組建檔案的版本詳細資訊。此資訊非常有用，因為每個遊戲伺服器的新版本都需要新的建置資源。
- storage-location
 - Bucket – 包含您建置的 S3 儲存貯體名稱。例如，"my_build_files"。
 - Key – 包含建置檔案的 .zip 檔案名稱。例如，「my_game_build_7.0.1、7.0.2」。

- RoleARN – 指派給您建立之 IAM 角色的 ARN。例如，「arn:aws:iam::111122223333:role/GameLiftAccess」。如需政策範例，請參閱 [在 Amazon S3 中存取遊戲組建檔案](#)。
- region – 在您計劃部署機群的 AWS 區域中建立組建。如果您要在多個區域中部署遊戲，請在每個區域中建立組建。

 Note

建議您使用 [configure get](#) 命令檢查目前的預設區域。若要變更您的預設區域，請使用 [configure set](#) 命令。

範例

```
aws gamelift create-build \  
  --operating-system WINDOWS_2016 \  
  --storage-location  
  "Bucket"="my_game_build_files","Key"="mygame_build_101.zip","RoleArn"="arn:aws:iam::111122223333:role/GameLiftAccess" \  
  gamelift" \  
  --name "My Game Nightly Build" \  
  --build-version "build 101" \  
  --region us-west-2
```

3. 若要檢視新組建，請使用 [describe-build](#) 命令。

更新Amazon GameLift Servers受管託管的遊戲伺服器建置

當您為Amazon GameLift Servers受管 EC2 託管部署遊戲伺服器組建時，您可以上傳遊戲伺服器軟體並建立Amazon GameLift Servers組建資源。建立Amazon GameLift Servers組建之後，您可以更新組建的中繼資料，但無法自行更新組建檔案。若要將更新部署到您的遊戲伺服器，請上傳更新的檔案，並使用 AWS CLI 命令[upload-build](#)建立新的Amazon GameLift Servers組建。或者，您可以使用[create-build](#)命令，從您控制的 Amazon S3 儲存貯體上傳新組建。然後，透過為其建立新的機群來部署新組建。

您可以更新組建的中繼資料，包括名稱和描述。對於這些任務，請使用 Amazon GameLift Servers主控台或 [update-build](#) AWS CLI 命令。

自動化遊戲建置更新

遵循這些提示，以協助自動化和簡化更新Amazon GameLift Servers受管機群遊戲伺服器組建的程序：

- 視需要使用遊戲工作階段佇列和交換機群。將遊戲工作階段請求傳送至 時Amazon GameLift Servers，請指定遊戲工作階段佇列，而非特定機群。使用佇列，您可以新增具有新建置的機群，並視需要移除舊機群。如需詳細資訊，請參閱[使用Amazon GameLift Servers佇列管理遊戲工作階段置放](#)。
- 使用別名將玩家轉移到新的遊戲組建。將遊戲工作階段請求傳送至 時Amazon GameLift Servers，請指定機群別名，而非機群 ID。如需詳細資訊，請參閱[建立Amazon GameLift Servers別名](#)。
- 設定 以進行反覆開發。在遊戲開發期間，探索設定支援快速反覆開發之託管測試環境的選項。請參閱 [使用 Amazon GameLift ServersAnywhere 設定反覆開發](#)。

請從 Github 上的 [Amazon GameLift Servers Toolkit](#) 嘗試這些資源：

快速建置更新工具（僅用於開發）

此工具可協助您修改已部署在受管 EC2 機群中運算上的遊戲伺服器組建，節省您在快速開發反覆運算期間的時間。此工具有數個選項；您可以取代整個遊戲組建或變更特定檔案，也可以管理如何在更新後重新啟動遊戲伺服器程序。您也可以使用它來更新機群中的所有運算，或鎖定個別運算。

請造訪 Github 中的 Amazon GameLift Servers Toolkit 儲存庫，以取得 Github 中的[快速建置更新工具](#)，並進一步了解如何使用它。

生產部署範例指令碼

此指令碼說明如何自動更新在生產環境中部署在受管 EC2 機群上的遊戲伺服器組建。若要使用此指令碼，您的Amazon GameLift Servers託管解決方案必須使用別名來抽象機群 IDs。範例指令碼會處理下列基本步驟：上傳更新的組建、建立新的組建並部署到新的機群、將玩家流量從現有機群重新導向到新的機群，以及刪除舊機群。自訂範例指令碼以符合您的特定部署需求。

請造訪 Github 中的 Amazon GameLift Servers Toolkit 儲存庫，以取得 Github 中的[生產部署範例指令碼](#)，並進一步了解如何使用。

建置 的容器映像 Amazon GameLift Servers

本主題說明如何使用遊戲伺服器軟體建立容器映像，以搭配 使用Amazon GameLift Servers。遊戲伺服器容器映像包含遊戲伺服器可執行檔及其執行所需的任何相依性。遊戲伺服器容器映像會與 Amazon GameLift Servers受管容器託管解決方案搭配使用。如需建置完整解決方案的詳細資訊，請參閱：

- [使用Amazon GameLift Servers受管容器託管的開發藍圖](#)
- [容器在 中的運作方式 Amazon GameLift Servers](#)

完成下列任務，讓您的遊戲伺服器容器映像準備好部署到 Amazon GameLift Servers 容器機群。開始這些任務之前，請先完成將您的遊戲伺服器程式碼與Amazon GameLift Servers伺服器 SDK 整合。

主題

- [建立遊戲伺服器容器映像](#)
- [將容器映像推送至 Amazon ECR](#)

建立遊戲伺服器容器映像

在 Linux 型平台上或使用已安裝 Docker 的適用於 Linux 的 Windows 子系統 (WSL) 時，請遵循這些指示。

建立遊戲伺服器容器映像

1. 使用遊戲伺服器軟體準備工作目錄。在本機電腦上，建立工作目錄來組織遊戲伺服器容器的檔案。容器映像將容器部署到Amazon GameLift Servers資源以進行託管時，會使用此檔案結構。例如：

```
[~/]$ mkdir -p work/glc/gamebuild && cd work && find .  
.  
./glc  
./glc/gamebuild
```

Note

如果您嘗試使用此功能，但還沒有有效的遊戲伺服器建置，請嘗試我們的範例遊戲伺服器 [SimpleServer](#)，其可在 GitHub 上取得。

2. 使用提供的範本建立新的 Dockerfile。
3. 請依照 Dockerfile 範本中的指示進行更新，以供您自己使用。
 - 視需要更新基礎映像。
 - 為您的遊戲伺服器建置設定環境變數。
4. 建置容器映像。執行 `docker build`，指定您自己的儲存庫名稱。例如：

```
[~/work/glc]$ docker build -t <local repository name>:<optional tag> .
```

您可以使用 `docker images` 命令檢視儲存庫和映像 IDs，如本範例所示：

遊戲伺服器容器映像的 Dockerfile 範本

此範本包含遊戲伺服器容器在機 Amazon GameLift Servers 群中可使用的最低指示。視需要修改遊戲伺服器的內容。

```
# Base image
# -----
# Add the base image that you want to use,
# Make sure to use an image with the same architecture as the
# Instance type you are planning to use on your fleets.
FROM public.ecr.aws/amazonlinux/amazonlinux
#
# Game build directory
# -----
# Add your gamebuild directory to the env variable below.
# The game build provided here needs to be integrated with server sdk for Amazon
GameLift Servers.
ENV GAME_BUILD_DIRECTORY="<ADD_GAME_BUILD_DIRECTORY>" \
#
# Game executable and launch parameters
# -----
# Add the relative path to your executable in the 'GAME_EXECUTABLE' env variable
below.
# The game build provided over here needs to be integrated with server sdk for Amazon
GameLift Servers.
# This template assumes that the executable path is relative to the game build
directory.
# Add any launch parameters to pass into your executable in the 'LAUNCH_PARAMS' env
variable below.
# Add 'HOME_DIR' to identify where the game executable and logs exist.
GAME_EXECUTABLE="<ADD NAME OF EXECUTABLE WITHIN THE GAME DIRECTORY>" \
LAUNCH_PARAMS=<ADD LAUNCH PARAMETERS> \
HOME_DIR="/local/game" \

# Install dependencies as necessary
RUN yum install -y shadow-utils
```

```
RUN mkdir -p $HOME_DIR
COPY ./${GAME_BUILD_DIRECTORY}/ $HOME_DIR

# Change directory to home
WORKDIR $HOME_DIR

# Set up for 'gamelift' user
RUN useradd -m gamelift && \
    echo "gamelift ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers && \
    chown -R gamelift:gamelift $HOME_DIR

# Add permissions to game build
RUN chmod +x ./${GAME_EXECUTABLE}

USER gamelift

# Check directory before starting the container
RUN ls -lhrt .

# Check path before starting the container
RUN echo $PATH

# Start the game build
ENTRYPOINT ["/bin/sh", "-c", "./${GAME_EXECUTABLE}", "${LAUNCH_PARAMS}]
```

將容器映像推送至 Amazon ECR

建立容器映像以部署至 之後 Amazon GameLift Servers，請將映像存放在 Amazon ECR 中的公有或私有儲存庫中。此儲存庫會獲指派 URI 值，其 Amazon GameLift Servers 會使用 擷取映像的快照，以部署至容器機群。

Note

如果您還沒有 Amazon ECR 私有儲存庫，請[建立一個](#)。

將容器映像推送至 Amazon ECR

1. 取得您的 Amazon ECR 登入資料。將容器映像推送至 Amazon ECR 之前，請先以暫時形式取得您的 AWS 登入資料，並將其提供給 Docker。Docker 需要這些登入資料才能登入。

```
[~/work/glc]$ aws ecr get-login-password --region us-west-2 | docker login --  
username AWS --password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com  
WARNING! Your password will be stored unencrypted in  
/home/user-name/.docker/config.json.  
Configure a credential helper to remove this warning.  
See https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
  
Login Succeeded
```

2. 複製您要使用的 [Amazon ECR 私有儲存庫](#) 的 URI。
3. 將 Amazon ECR 標籤套用至您的容器映像。

Example

```
[~/work/glc]$ docker tag <IMAGE ID from above> <Amazon ECR private repository  
URI>:<optional tag>
```

4. 將您的容器映像推送至 Amazon ECR

Example

```
[~/work/glc]$ docker image push <Amazon ECR private repository URI>
```

部署 的指令碼 Amazon GameLift ServersRealtime

當您準備好部署遊戲時，請將完成Amazon GameLift ServersRealtime的Realtime伺服器指令碼檔案上傳至 Amazon GameLift Servers。透過建立Amazon GameLift Servers指令碼資源並指定指令碼檔案的位置來執行此操作。您也可以透過上傳現有指令碼資源的新檔案來更新已部署的伺服器指令碼檔案。

當您建立新的指令碼資源時，會為其Amazon GameLift Servers指派唯一的指令碼 ID（例如 `script-1111aaaa-22bb-33cc-44dd-5555eeee66ff`），並上傳指令碼檔案的副本。上傳時間取決於指令碼檔案的大小和連線速度。

建立指令碼資源後，Amazon GameLift Servers 會使用新的Amazon GameLift ServersRealtime機群部署指令碼。會將您的伺服器指令碼Amazon GameLift Servers安裝到機群中的每個執行個體，並將指令碼檔案放在 `/local/game`。

若要疑難排解與伺服器指令碼相關的機群啟用問題，請參閱 [偵錯 Amazon GameLift Servers 機群問題](#)。

套件指令碼檔案

您的伺服器指令碼可以包含一個或多個合併為單一 .zip 檔案的檔案，以供上傳。 .zip 檔案必須包含指令碼需要執行的所有檔案。

您可以將壓縮的指令碼檔案存放在本機檔案目錄或 Amazon Simple Storage Service (Amazon S3) 儲存貯體中。

從本機目錄上傳指令碼檔案

如果您的指令碼檔案存放在本機，您可以從 Amazon GameLift Servers 該處將其上傳到。若要建立指令碼資源，請使用 Amazon GameLift Servers 主控台或 [AWS Command Line Interface \(AWS CLI\)](#)。

Amazon GameLift 伺服器 console

建立指令碼資源

1. 開啟 [Amazon GameLift Servers 主控台](#)。
2. 在導覽窗格中，選擇託管、指令碼。
3. 在指令碼頁面上，選擇建立指令碼。
4. 在建立指令碼頁面的指令碼設定下，執行下列動作：
 - a. 在名稱中，輸入指令碼名稱。
 - b. （選用）對於版本，輸入版本資訊。由於您可以更新指令碼的內容，版本資料有助於追蹤更新。
 - c. 針對指令碼來源，選擇上傳 .zip 檔案。
 - d. 對於指令碼檔案，選擇選擇檔案，瀏覽包含指令碼的 .zip 檔案，然後選擇該檔案。
5. （選用）在標籤下，輸入索引鍵和值對，將標籤新增至指令碼。
6. 選擇建立。

Amazon GameLift Servers 會將 ID 指派給新的指令碼，並上傳指定的 .zip 檔案。您可以在指令碼頁面上檢視新的指令碼，包括其狀態。

AWS CLI

使用 [create-script](#) AWS CLI 命令來定義新的指令碼，並上傳您的伺服器指令碼檔案。

建立指令碼資源

1. 將 .zip 檔案放入您可以使用的目錄 AWS CLI。
2. 開啟命令列視窗，並切換到您放置 .zip 檔案的目錄。
3. 輸入下列create-script命令和參數。對於 --zip-file 參數，請務必將字串新增至 .zip 檔案fileb://的名稱。它會將檔案識別為二進位，以便 Amazon GameLift Servers處理壓縮的內容。

```
aws gamelift create-script \  
  --name user-defined name of script \  
  --script-version user-defined version info \  
  --zip-file fileb://name of zip file \  
  --region region name
```

範例

```
aws gamelift create-script \  
  --name "My_Realtime_Server_Script_1" \  
  --script-version "1.0.0" \  
  --zip-file fileb://myrealtime_script_1.0.0.zip \  
  --region us-west-2
```

為了回應您的請求，會Amazon GameLift Servers傳回新的指令碼物件。

4. 若要檢視新的指令碼，請呼叫 [describe-script](#)。

從 Amazon S3 上傳指令碼檔案

您可以將指令碼檔案存放在 Amazon S3 儲存貯體中，並從Amazon GameLift Servers該處將其上傳至。建立指令碼時，您可以指定 S3 儲存貯體位置，並從 Amazon S3 Amazon GameLift Servers擷取指令碼檔案。

建立指令碼資源

1. 將指令碼檔案存放在 S3 儲存貯體中。建立包含伺服器指令碼檔案的 .zip 檔案，並將其上傳至您控制之 中的 S3 AWS 帳戶 儲存貯體。請記下物件 URI - 建立Amazon GameLift Servers指令碼時需要此項目。

 Note

Amazon GameLift Servers 不支援從名稱包含句點 (.) 的 S3 儲存貯體上傳。

2. 允許 Amazon GameLift Servers 存取您的指令碼檔案。若要建立允許 Amazon GameLift Servers 存取包含伺服器指令碼之 S3 儲存貯體的 AWS Identity and Access Management (IAM) 角色，請遵循 [中的指示設定 的 IAM 服務角色 Amazon GameLift Servers](#)。建立新角色之後，請記下建立指令碼時所需的名稱。
3. 建立指令碼。使用 Amazon GameLift Servers 主控台或 AWS CLI 建立新的指令碼記錄。若要提出此請求，您必須擁有 IAM PassRole 許可，如中所述的 [IAM 許可範例 Amazon GameLift Servers](#)。

Amazon GameLift 伺服器 console

1. 在 [Amazon GameLift Servers 主控台](#) 的導覽窗格中，選擇託管、指令碼。
2. 在指令碼頁面上，選擇建立指令碼。
3. 在建立指令碼頁面的指令碼設定下，執行下列動作：
 - a. 在名稱中，輸入指令碼名稱。
 - b. （選用）對於版本，輸入版本資訊。由於您可以更新指令碼的內容，版本資料有助於追蹤更新。
 - c. 針對指令碼來源，選擇 Amazon S3 URI。
 - d. 輸入您上傳到 Amazon S3 之指令碼物件的 S3 URI，然後選擇物件版本。Amazon S3 如果您不記得 Amazon S3 URI 和物件版本，請選擇瀏覽 S3，然後搜尋指令碼物件。
4. （選用）在標籤下，輸入金鑰和值對，將標籤新增至指令碼。
5. 選擇建立。

Amazon GameLift Servers 會將 ID 指派給新的指令碼，並上傳指定的 .zip 檔案。您可以在指令碼頁面上檢視新的指令碼，包括其狀態。

AWS CLI

使用 [create-script](#) AWS CLI 命令來定義新的指令碼，並上傳您的伺服器指令碼檔案。

1. 開啟命令列視窗，並切換到您可以使用的目錄 AWS CLI。

- 輸入下列create-script命令和參數。--storage-location 參數指定指令碼檔案的 Amazon S3 儲存貯體位置。

```
aws gamelift create-script \  
  --name [user-defined name of script] \  
  --script-version [user-defined version info] \  
  --storage-location "Bucket"=S3 bucket name,"Key"=name of zip file in S3 bucket,"RoleArn"=Access role ARN \  
  --region region name
```

範例

```
aws gamelift create-script \  
  --name "My_Realtime_Server_Script_1" \  
  --script-version "1.0.0" \  
  --storage-location "Bucket"="gamelift-script","Key"="myrealtime_script_1.0.0.zip","RoleArn"="arn:aws:iam::123456789012:role/S3Access" \  
  --region us-west-2
```

為了回應您的請求，會Amazon GameLift Servers傳回新的指令碼物件。

- 若要檢視新的指令碼，請呼叫 [describe-script](#)。

更新Amazon GameLift ServersRealtime指令碼檔案

您可以使用 Amazon GameLift Servers主控台或 [update-script](#) AWS CLI 命令來更新指令碼資源的中繼資料。

您也可以更新指令碼資源的指令碼內容。Amazon GameLift Servers 會將指令碼內容部署到使用更新指令碼資源的所有機群執行個體。部署更新後的指令碼時，執行個體會在啟動新的遊戲工作階段時使用它。在更新時已在執行的遊戲工作階段不會使用更新的指令碼。

更新指令碼檔案

- 對於本機存放的指令碼檔案，若要上傳更新的指令碼 .zip 檔案，請使用 Amazon GameLift Servers 主控台或 update-script命令。
- 對於存放在 Amazon S3 儲存貯體中的指令碼檔案，請將更新的指令碼檔案上傳到 S3 儲存貯體。會 Amazon GameLift Servers定期檢查更新的指令碼檔案，並直接從 S3 儲存貯體擷取它們。

使用 設定託管機群 Amazon GameLift Servers

在本節中，您將找到有關設計、建置和維護機Amazon GameLift Servers群以託管遊戲伺服器的資訊。請參閱 [Amazon GameLift Servers 託管選項](#) 以進一步了解 Amazon GameLift Servers提供的託管解決方案，包括使用受管 EC2 機群、內部部署硬體的自我管理 Anywhere 機群，以及同時使用兩者的混合解決方案。

主題

- [機群特性](#)
- [Amazon GameLift Servers 機群建立的運作方式](#)
- [Amazon GameLift Servers 受管 EC2 機群](#)
- [Amazon GameLift Servers 受管容器機群](#)
- [Amazon GameLift Servers Anywhere 機群](#)
- [使用別名抽象機Amazon GameLift Servers群指定](#)

機群特性

Amazon GameLift Servers 機群是一組運算資源，可為玩家執行遊戲伺服器和託管遊戲工作階段。機群可能因您使用的運算資源類型以及機群的管理方式而有所不同。機群的大小 - 可以支援的遊戲工作階段和玩家數量 - 取決於您提供的運算資源數量。所有機Amazon GameLift Servers群都有下列特性：

- 在所有機群上執行的遊戲伺服器程序會與適用於 的伺服器 SDK Amazon GameLift Servers 整合，Amazon GameLift Servers並以相同方式與服務通訊。遊戲伺服器報告其託管遊戲工作階段和玩家的可用性、回應啟動或停止遊戲工作階段的提示，以及其他互動。
- Amazon GameLift Servers 會以相同方式處理所有機群的遊戲工作階段置放。Amazon GameLift Servers會追蹤機群的遊戲伺服器狀態，並從可用的遊戲伺服器中選擇以託管新的遊戲工作階段。無論您的遊戲將遊戲工作階段放置在單一機群，或使用[遊戲工作階段佇列](#)來平衡多個機群的託管，都會使用此程序。透過佇列，您也可以自訂置放決策，以考慮資源成本和延遲等因素。
- 所有機群都支援與遊戲工作階段置放佇列合作使用FlexMatch配對建構器。Amazon GameLift Servers 服務會接收玩家配對請求、形成配對，並將它們傳遞至遊戲工作階段佇列，以尋找可用的遊戲伺服器。
- Amazon GameLift Servers 會收集各種機群指標。這些包括運算和伺服器程序的狀態指標，以及遊戲工作階段和玩家活動的用量指標。如需可用指標的完整清單，請參閱 [使用 Amazon CloudWatch 監控 Amazon GameLift Servers](#)。

Amazon GameLift Servers 機群建立的運作方式

當您請求新的機群時，會Amazon GameLift Servers啟動工作流程來建立機群資源。當它完成工作流程的每個步驟時，會Amazon GameLift Servers更新機群的狀態並發出一系列事件，以傳達機群建立的進度。

Amazon GameLift Servers 使用兩種類型的事件。機群狀態轉換事件會在機群狀態變更時標記。機群建立事件提供額外的標記，以協助偵錯建立問題。您可以使用 Amazon GameLift Servers主控台或呼叫 Amazon GameLift Servers API 操作 [DescribeFleetEvents](#) 來追蹤所有事件。您也可以使用 [DescribeFleetAttributes](#) 或 [DescribeFleetLocationAttributes](#) 追蹤機群和位置狀態。

Amazon GameLift Servers 受管 EC2 機群

Amazon GameLift Servers 受管 EC2 機群為生產託管提供雲端型資源。透過受管機群，您可以取得針對多玩家遊戲託管最佳化之 AWS 雲端 資源的彈性、安全性和可靠性。Amazon GameLift Servers提供強大的主機管理工具。

受管 EC2 機群是一組 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，其Amazon GameLift Servers擁有並根據您的組態運作。這些執行個體實際位於支援的 AWS 區域 或 Local Zones。建立機群時，您可以選擇符合遊戲伺服器對運算能力、記憶體、儲存體、聯網功能需求的 EC2 執行個體類型。

啟動機群中的每個執行個體時，Amazon GameLift Servers 會使用所需的執行時間環境部署您的遊戲伺服器組建。執行期環境使用建立機群時可用的最新 Amazon Machine Image (AMI) 版本。機群中的所有執行個體都使用相同的 AMI 版本。

Note

根據最佳實務，建議您每 30 天替換一次機群，以維護託管遊戲伺服器的安全up-to-date執行期環境。這需要建立新的機群，並將玩家流量遷移到其中。如需詳細的指導方針，請參閱 [Amazon GameLift Servers 的安全最佳實務](#)。

在執行個體上安裝執行期環境和遊戲伺服器建置之後，會Amazon GameLift Servers開始啟動遊戲伺服器程序。每個遊戲伺服器程序都會建立 Amazon GameLift Servers服務的連線、報告託管遊戲工作階段的準備程度，並開始通訊運作狀態。然後Amazon GameLift Servers可以提示伺服器程序啟動遊戲工作階段。

除了機群部署之外，還會Amazon GameLift Servers處理下列主機管理任務，因此您不需要：

- 追蹤機群中所有運算的狀態，並取代過時或運作狀態不佳的運算。
- 處理伺服器程序與服務之間通訊的身分驗證Amazon GameLift Servers。
- 根據您的執行時間組態，自動在每個運算上啟動和停止遊戲伺服器程序。
- 提供容量擴展工具，包括自動擴展，可動態調整機群容量以滿足玩家需求。
- 報告機群 EC2 執行個體的效能指標。

請參閱這些主題，了解如何設定和維護受管 EC2 機群：

- [使用Amazon GameLift Servers受管 EC2 託管的開發藍圖](#)
- [建立Amazon GameLift Servers受管 EC2 機群](#)
- [使用 擴展遊戲託管容量 Amazon GameLift Servers](#)
- [自訂您的 Amazon GameLift Servers EC2 受管機群](#)
- [更新Amazon GameLift Servers機群組態](#)

受管 EC2 機群建立工作流程

對於受管機群，Amazon GameLift Servers會設定機群資源，並在安裝和執行遊戲伺服器軟體的情況下部署一組運算資源。當建立工作流程完成且成功時，機群在機群主區域中有一個作用中的 EC2 執行個體，並在機群的遠端位置各有一個執行個體。所有執行個體都已準備好託管遊戲工作階段。

1. Amazon GameLift Servers 在機群的主區域中建立機群資源，並將每個位置所需的容量設定為一 (1) 個執行個體。機群和位置狀態設定為新增。
2. Amazon GameLift Servers 開始將事件寫入機群事件日誌。
3. Amazon GameLift Servers 會將機群狀態設定為下載，並開始準備要部署的遊戲伺服器軟體。
 - a. 取得上傳的遊戲伺服器建置並擷取壓縮檔案。
 - b. 如果提供，則執行安裝指令碼。
 - c. 將機群狀態設定為驗證，並開始驗證下載和安裝組建檔案時未發生錯誤。
4. Amazon GameLift Servers 會將機群狀態設定為建置、設定機群硬體，以及為每個機群執行個體配置一個 EC2 執行個體。
5. Amazon GameLift Servers 會將機群狀態設定為啟用。在每個執行個體上啟動遊戲伺服器程序（根據機群的執行時間指示），並測試組建Amazon GameLift Servers與服務之間的連線。
6. 當遊戲伺服器在每個執行個體上處理連線並報告準備託管遊戲工作階段時，Amazon GameLift Servers 會將機群和位置狀態設定為作用中。此時，機群被視為已準備好託管遊戲工作階段。

建立Amazon GameLift Servers受管 EC2 機群

本主題說明如何建立Amazon GameLift Servers受管 EC2 機群。受管機群使用針對多玩家遊戲託管最佳化的 Amazon Elastic Compute Cloud (Amazon EC2) 運算執行個體。您可以建立受管機群，將運算部署到 全域至 AWS 區域 和 支援的 Local ZonesAmazon GameLift Servers。

當您建立新的受管 EC2 機群資源時，您會立即啟動機群建立的第一階段。當 部署 EC2 Amazon GameLift Servers 執行個體、在執行個體上安裝執行期環境和遊戲伺服器組建，並開始啟動遊戲伺服器時，受管機群建立會經過數個階段。根據遊戲伺服器建置所需的執行時間環境，Amazon GameLift Servers 會在機群建立時部署最新版本的 Amazon Machine Image (AMI) (且機群中的所有未來執行個體將使用相同的版本)。您可以在 主控台 或使用 AWS Command Line Interface () 監控機群的狀態 AWS CLI。當機群準備好託管遊戲工作階段時，狀態會變更為 ACTIVE。如需受管機群建立的詳細資訊，請參閱下列主題：

- [Amazon GameLift Servers 機群建立的運作方式](#)
- [偵錯 Amazon GameLift Servers 機群問題](#)

Note

根據最佳實務，建議您每 30 天替換一次機群，以維護託管遊戲伺服器的安全up-to-date執行期環境。這需要建立新的機群，並將玩家流量遷移到其中。如需詳細的指導方針，請參閱 [Amazon GameLift Servers 的安全最佳實務](#)。

建立受管 EC2 機群

使用 Amazon GameLift Servers主控台或 AWS Command Line Interface (AWS CLI) 來建立受管 EC2 機群。

Console

在 [Amazon GameLift Servers主控台](#) 中，使用導覽窗格開啟機群頁面。選擇建立機群以開始機群建立工作流程。

步驟 1 選擇運算類型

選取受管 EC2 選項，然後選擇下一步。

步驟 2 定義機群詳細資訊

在此步驟中，指定一些整個機群的設定。

i 如需最少的機群組態：

- 提供機群名稱。
- 選擇二進位類型，並指定上傳的組建或指令碼。
- 略過其他詳細資訊和標籤的區段。

1. 填寫機群詳細資訊區段：

- a. 輸入機群名稱。我們建議您使用機群命名模式，以便在檢視機群清單時更容易識別機群類型。
- b. 提供機群的簡短描述。
- c. 針對二進位類型，選取建置以指出您要部署自訂遊戲伺服器建置，或者如果您要部署 Amazon GameLift Servers Realtime 到此機群，請選取 或 指令碼。從下拉式清單中選取上傳的組建或指令碼。

2. (選用) 視需要設定其他詳細資訊。

- a. 如果您的遊戲伺服器可執行檔需要存取帳戶中的其他 AWS 資源，請指定具有必要許可的 IAM 執行個體角色。如需詳細資訊，包括如何授權其他伺服器端應用程式（例如 CloudWatch 代理程式），請參閱 [與機群中的其他 AWS 資源通訊](#)。在您建立機群之後，就無法變更此設定。

您必須先建立角色，才能建立使用該角色的機群。此外，若要使用執行個體角色建立機群，您的 AWS 使用者必須具有 IAM PassRole 許可（請參閱 [的 IAM 許可範例 Amazon GameLift Servers](#)）。

- b. 開啟產生 TLS 憑證選項來設定遊戲的身分驗證和加密，遊戲用戶端會在連線和加密所有用戶端/伺服器通訊時使用此憑證來驗證遊戲伺服器。對於已啟用 TLS 之機群中的每個執行個體，Amazon GameLift Servers 也會使用憑證來建立新的 DNS 項目。在您建立機群之後，就無法變更此設定。
- c. 如果您想要結合此機群和其他機群的指標資料，請指定指標群組名稱。針對您要合併的所有機群，使用相同的指標群組名稱。檢視指標群組的指標，以查看彙總的資料。

3. (選用) 將標籤新增至機群資源。每個標籤皆包含由您定義的一個金鑰與一個選用值。將標籤指派給您想要以實用方式分類 AWS 的資源，例如依用途、擁有者或環境。為您要新增的每個標籤選擇新增標籤。
4. 選擇下一步以繼續工作流程。

步驟 3 定義執行個體詳細資訊

在此步驟中，指定要使用的託管資源類型，以及您要部署資源的位置。透過選擇多個位置，您可以將遊戲伺服器部署到更寬廣的地理位置，讓它們更接近您的玩家，並將延遲降至最低。並非所有 EC2 執行個體類型都可在所有位置使用。

如需最少的機群組態：

- 請勿新增遠端位置。
- 將機群類型設定為「隨需」。Spot 機群需要額外的設定工作。
- 將執行個體類型設定為 "c5.large"。此常用執行個體類型適用於所有 AWS 區域。

1. 在執行個體部署中，指定機群位置和類型。
 - a. 選取一或多個您要部署機群執行個體的其他位置。這些遠端位置會新增至機群的主位置（預先選取），這是您要建立此機群 AWS 區域的。您可以從 Amazon GameLift Servers 支援的所有 AWS 區域和 Local Zones 中選取遠端位置。

若要進一步了解支援的位置，包括如何使用預設未啟用 AWS 區域的，請參閱 [Amazon GameLift Servers 服務位置](#) 以取得受管託管。同時檢閱每個機群位置的 Amazon GameLift Servers [配額](#)。
 - b. 選擇使用此機群的隨需或 Spot 執行個體。如需機群類型的詳細資訊，請參閱 [隨需執行個體與 Spot 執行個體](#)。
2. 選擇符合您需求的 Amazon EC2 執行個體組態，並且可在所有選取的位置使用。此清單會根據您目前的位置和機群類型選擇進行篩選。您可以根據執行個體類型系列和架構等其他因素進一步篩選。建立機群後，您無法變更執行個體類型。

有些位置的執行個體類型選項有限。如果您偏好的執行個體類型不適用於所有位置，請選擇位置可用性值以檢視完整詳細資訊。若要容納所有位置，您可能需要建立具有不同執行個體類型的個別機群。

如需選擇執行個體類型的詳細資訊，請參閱 [執行個體類型](#)。若要進一步了解 Amazon EC2 Arm 架構，請參閱 [AWS Graviton 處理器](#) 和 [Amazon EC2 執行個體類型](#)。如需支援的執行個體類型完整清單 Amazon GameLift Servers，請參閱 [EC2InstanceType](#) () 的 API 參考 [CreateFleet\(\)](#)。

 Note

Graviton Arm 執行個體需要在 Linux 作業系統上建置 Amazon GameLift Servers 伺服器。C++ 和 C# 需要伺服器 SDK 5.1.1 或更新版本。Go 需要伺服器 SDK 5.0 或更新版本。這些執行個體在 out-of-the-box 可用支援。AL2023AL2

3. 選擇下一步以繼續工作流程。

步驟 4 設定執行時間

在此步驟中，描述您希望機群中的每個執行個體如何執行遊戲伺服器軟體。為每個要在執行個體上執行的可執行檔定義個別伺服器程序明細項目，並決定要同時執行多少個伺服器程序。在每個執行個體上開啟連接埠，以允許玩家直接連線到遊戲伺服器。您可以隨時更新這些機群設定。

 如需最少的機群組態：

- 為您的遊戲伺服器可執行檔定義單一伺服器程序明細項目。如果您的遊戲伺服器需要執行其他程序，也請為每個程序建立定義。
- 為每個明細項目使用預設的並行程序數目 (1)。
- 略過遊戲工作階段啟用設定。
- 指定單一連接埠號碼。
- 略過遊戲工作階段資源設定。

1. 建立執行期組態，指示 Amazon GameLift Servers 如何在機群中的每個執行個體上執行伺服器程序。您可以在部署之後隨時變更機群的執行時間組態。
 - a. 輸入建置中可執行檔的啟動路徑。在 Windows 執行個體上，遊戲伺服器可執行檔會建置到路徑 `C:\game`。在 Linux 執行個體上，遊戲伺服器是建置到 `/local/game`。範例：`C:\game\MyGame\server.exe`、`/local/game/MyGame/server.exe` 或 `MyRealtimeLaunchScript.js`。

- b. 輸入選用的啟動參數以傳遞至您的遊戲可執行檔。範例：**+sv_port 33435 +start_lobby**。
 - c. 指定在每個執行個體上執行的並行程序數目。對於遊戲伺服器可執行檔，每個程序可以託管一個遊戲工作階段，因此並行程序會決定執行個體可以同時託管的遊戲工作階段數量。

檢閱每個執行個體伺服器程序的Amazon GameLift Servers[配額](#)。這些配額適用於所有組態的總並行程序。如果您將機群設定為超過它們，則機群無法啟用。
2. 使用遊戲工作階段啟用預設值或為您的遊戲自訂它們。如果執行時間組態呼叫每個執行個體的多個並行遊戲伺服器程序，這些設定會決定新遊戲工作階段的啟動速度。
 - a. 設定並行遊戲工作階段啟用上限，以限制執行個體上準備新遊戲工作階段的遊戲伺服器數量。此設定在啟動多個新遊戲工作階段時非常耗費資源，且可能會影響其他執行中遊戲工作階段的效能。
 - b. 設定新的啟用逾時，以反映新遊戲工作階段完成啟用並回報準備好託管玩家所需的時間上限。如果超過此值，會Amazon GameLift Servers終止遊戲工作階段啟用。
3. 開啟 EC2 連接埠設定，以允許傳入流量存取機群上的伺服器程序。建立機群不需要這些設定，但您需要先設定這些設定，玩家才能連線到機群上的遊戲工作階段。

針對每個連接埠設定，選擇用於遊戲用戶端和遊戲伺服器之間通訊的資料傳輸通訊協定類型。使用 CIDR 表示法提供連接埠範圍（格式為 nnnnn[-nnnnn]）和 IP 地址範圍 **0.0.0.0/0**（例如允許任何人存取）。

如果您需要設定多個非連續範圍，請建立多個連接埠設定。
4. 指定選用的遊戲工作階段資源設定。您可以在部署之後隨時更新這些設定。
 - a. 開啟或關閉機群中所有執行個體的遊戲擴展保護政策。在縮減規模事件期間，如果受保護的機群執行個體託管作用中的遊戲工作階段，Amazon GameLift Servers則不會終止這些執行個體。
 - b. 如果您想要限制一個玩家可在指定時間範圍內建立的遊戲工作階段數量，請設定資源建立上限。
5. 選擇下一步以繼續工作流程。

步驟 5 檢閱和建立

建立機群之前，請先檢閱您的設定。雖然某些設定可以稍後更新（請參閱 [更新Amazon GameLift Servers機群組態](#)），但在建立機群之後，不允許對下列設定進行變更：

- **運算類型**：您無法將受管 EC2 機群轉換為 Anywhere 機群。
- **組建或指令碼**：若要將更新部署到您的遊戲伺服器組建或指令碼，您必須建立新的機群。
- **其他選項**，包括執行個體角色和產生 TLS 憑證。
- **執行個體詳細資訊**，包括機群類型 (Spot 或隨需) 和 EC2 執行個體類型。

當您準備好部署新機群時，請選擇建立。會Amazon GameLift Servers立即開始機群啟用程序，指派唯一的 ID 並將機群置於 NEW 狀態。從機群頁面追蹤機群進度。查看機群的詳細資訊頁面，然後前往事件索引標籤。

您可以在機群達到 ACTIVE 狀態後調整機群的託管容量。Amazon GameLift Servers 最初會在每個機群位置部署具有單一執行個體的機群。並透過將執行個體新增至每個位置來調整容量。如需詳細資訊，請參閱[使用 擴展遊戲託管容量 Amazon GameLift Servers](#)。

AWS CLI

使用 `create-fleet` 命令來建立運算類型 的機群EC2。會在您目前的預設值中Amazon GameLift Servers建立機群資源 AWS 區域 (或者您可以新增 `--region` 標籤來指定不同的 AWS 區域)。

建立最少的受管機群

下列範例請求會使用部署機群所需的最低設定建立新的機群，該機群具有遊戲用戶端可以連線的執行中遊戲伺服器。新機群具有下列特性：

- 它指定遊戲伺服器組建，已上傳至 Amazon GameLift Servers 並處於 READY 狀態。
- 使用 c5.large 隨需執行個體搭配符合所選遊戲組建的作業系統。
- 它將機群的首頁設定為 AWS 區域 us-west-2，並僅將執行個體部署到該區域。
- 根據執行時間組態，機群中的每個運算都會執行一個遊戲伺服器程序，這表示每個運算一次只能託管一個遊戲工作階段。遊戲工作階段啟用逾時設定為預設值 300 秒，同時啟用次數沒有限制。
- 玩家可以使用 的單一連接埠設定連接到遊戲伺服器 33435。
- 所有其他功能都會關閉或使用預設設定。

```
aws gamelift create-fleet \
  --name MinimalFleet123 \
  --description "A basic test fleet" \
  --region us-west-2 \
  --ec2-instance-type c5.large \
  --fleet-type ON_DEMAND \
```

```
--build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \
--runtime-configuration "ServerProcesses=[{LaunchPath=C:\game\Bin64.dedicated
\MultiplayerSampleProjectLauncher_Server.exe, ConcurrentExecutions=10}]" \
--ec2-inbound-permissions
"FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
```

建立完全設定的受管機群

下列範例請求會使用所有選用功能的設定來建立生產機群。新機群具有下列特性：

- 它指定遊戲伺服器組建，已上傳至 Amazon GameLift Servers 並處於 READY 狀態。
- 它使用 c5.large 隨需執行個體搭配符合所選遊戲組建的作業系統。
- 它將機群的首頁設定為 AWS 區域 us-west-2，並將執行個體部署到主要區域 和一個遠端位置 sa-east-1。
- 根據執行時間組態：
 - 機群中的每個運算會使用相同的啟動參數執行 10 個遊戲伺服器程序，這表示每個運算最多可同時託管 10 個遊戲工作階段。
 - 每次運算時，只能同時啟用兩個遊戲工作階段。啟用遊戲工作階段必須準備好在 300 秒 (5 分鐘) 內託管玩家或終止。
- 玩家可以使用下列範圍的連接埠 連線到遊戲伺服器 33435 to 33535。
- 它會產生 TLS 憑證，用於遊戲用戶端和伺服器之間的加密通訊。
- 機群中的所有遊戲工作階段都已開啟遊戲工作階段保護。
- 個別玩家僅限於在 15 分鐘內建立三個新的遊戲工作階段。
- 此機群的指標包含在指標群組 中 AMERfleets，(在此範例中) 會彙總北美洲、中南美洲機群群組的指標。

```
aws gamelift create-fleet \
  --name ProdFleet123 \
  --description "A fully configured prod fleet" \
  --ec2-instance-type c5.large \
  --region us-west-2 \
  --locations "Location=sa-east-1" \
  --fleet-type ON_DEMAND \
  --build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \
  --certificate-configuration "CertificateType=GENERATED" \
  --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
  MaxConcurrentGameSessionActivations=2, ServerProcesses=[{LaunchPath=C:\game
```

```
\Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe, Parameters="+sv_port
33435 +start_lobby, ConcurrentExecutions=10}]" \
  --new-game-session-protection-policy "FullProtection" \
  --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
PolicyPeriodInMinutes=15" \
  --ec2-inbound-permissions
"FromPort=33435,ToPort=33535,IpRange=0.0.0.0/0,Protocol=UDP" \
  --metric-groups "AMERfleets"
```

如果建立機群請求成功，會Amazon GameLift Servers傳回一組機群屬性，其中包含您請求的組態設定和新的機群 ID。Amazon GameLift Servers然後啟動機群啟用程序，並將機群狀態和位置狀態設定為新增。您可以追蹤機群的狀態，並使用這些 CLI 命令檢視其他機群資訊：

- [describe-fleet-events](#)
- [describe-fleet-attributes](#)
- [describe-fleet-capacity](#)
- [describe-fleet-port-settings](#)
- [describe-fleet-utilization](#)
- [describe-runtime-configuration](#)
- [describe-fleet-location-attributes](#)
- [describe-fleet-location-capacity](#)
- [describe-fleet-location-utilization](#)

您可以使用以下命令，視需要變更叢集的容量及其他組態設定：

- [update-fleet-attributes](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [update-runtime-configuration](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

自訂您的 Amazon GameLift Servers EC2 受管機群

本節中的主題概述在建置具有Amazon GameLift Servers受管 EC2 機群的託管解決方案時，您可以進行的一些自訂。它們提供建立和設定機群以託管遊戲的指引和最佳實務。

必要的決策包括：

- 您應該在何處部署託管資源？延遲是選取機群位置的主要因素，但成本也會因位置而異。
- 哪種 EC2 執行個體類型最能支援您的遊戲？從所有機群位置可用的執行個體類型中選擇，以使用運算架構、記憶體、儲存和聯網容量的最佳組合。
- 您需要多少大小的執行個體類型？根據遊戲伺服器軟體的資源需求（記憶體和 CPU）和其他因素，選擇執行個體類型大小。
- 您的機群是否應該使用隨需或 Spot 執行個體？考慮您是否可以利用較低的 Spot Amazon GameLift Servers 定價，因為如何防止遊戲工作階段中斷的機會。
- 您希望遊戲伺服器軟體如何在每個機群執行個體上執行？執行時間組態會告知要執行Amazon GameLift Servers的伺服器軟體及其方式。

主題

- [選擇受管機群的運算資源](#)
- [管理 如何Amazon GameLift Servers啟動遊戲伺服器](#)

選擇受管機群的運算資源

若要在雲端部署遊戲伺服器和託管遊戲工作階段，Amazon GameLift Servers提供使用稱為執行個體之[Amazon Elastic Compute Cloud \(Amazon EC2\) 資源](#)的受管機群。使用下列主題來協助決定您要用於受管託管解決方案的 EC2 執行個體類型，以及如何設定它們來執行遊戲伺服器軟體。

Note

如果您計劃使用您擁有的託管資源，無論是內部部署硬體或其他雲端型託管，請考慮使用 Amazon GameLift ServersAnywhere 進行混合託管的選項。請參閱 [使用 設定託管機群 Amazon GameLift Servers](#)。

主題

- [機群位置](#)
- [隨需執行個體與 Spot 執行個體](#)
- [作業系統](#)
- [執行個體類型](#)
- [Service Quotas](#)

機群位置

考慮您計劃部署遊戲伺服器的地理位置。執行個體類型可用性會因 AWS 區域 和 Local Zone 而有所不同。

對於多位置機群，執行個體可用性和配額取決於機群主要區域和所選遠端位置的組合。如需機群位置的詳細資訊，請參閱 [Amazon GameLift Servers 服務位置](#)。

考慮使用 [UDP ping 信標](#) 收集不同地理位置的網路延遲資料，以預測玩家裝置和潛在機群位置之間的延遲。這些特殊端點接受 UDP 訊息，而不是傳統的 ICMP ping，提供準確的延遲測量，以協助您選擇最佳的機群位置。

對於 Amazon GameLift ServersAnywhere 機群，您可以判斷實體硬體的位置。如需自訂位置的詳細資訊，請參閱 [Amazon GameLift ServersAnywhere 的位置](#)。

隨需執行個體與 Spot 執行個體

Amazon EC2 隨需執行個體和 Spot 執行個體提供相同的硬體和效能，但可用性和成本有所不同。

隨需執行個體

您可以在需要時取得隨需執行個體，並隨需保留。隨需執行個體具有固定成本，這表示您需要支付使用時間，而且沒有長期承諾。

Spot 執行個體

Spot 執行個體可以利用未使用的 AWS 運算容量，為隨需執行個體提供經濟實惠的替代方案。Spot 執行個體價格會根據每個位置中每個執行個體類型的供需而波動。AWS 可以在需要恢復容量時中斷 Spot 執行個體。Amazon GameLift Servers 會使用佇列和 FleetIQ 演算法來判斷 AWS 將中斷 Spot 執行個體，使執行個體進入回收狀態。然後，當執行個體上沒有作用中的遊戲工作階段時，Amazon GameLift Servers 會嘗試取代它。

如需如何使用 Spot 執行個體的詳細資訊，請參閱 [設計 Spot 執行個體的佇列](#)。

作業系統

Amazon GameLift Servers 執行個體支援在 Microsoft Windows 或 Amazon Linux 上執行的遊戲伺服器建置。當您將遊戲組建上傳至 Amazon GameLift Servers，請指定遊戲的作業系統。當您建立 Amazon EC2 機群來部署遊戲組建時，Amazon GameLift Servers 會自動使用組建的作業系統設定執行個體。如需支援的遊戲伺服器作業系統的詳細資訊，請參閱 [取得 Amazon GameLift Servers 開發工具](#)。

使用 Amazon GameLift Servers Anywhere 機群時，您可以使用硬體支援的任何作業系統。Amazon GameLift Servers Anywhere 機群會要求您將遊戲組建部署到硬體，同時使用 Amazon GameLift Servers 在一個位置管理您的資源。

執行個體類型

Amazon EC2 機群的執行個體類型會決定執行個體使用的硬體類型。不同的執行個體類型提供不同的運算能力、記憶體、儲存和聯網功能組合。

在為您的遊戲選擇可用的執行個體類型時，請考慮：

- 遊戲伺服器的運算架構：x64 或 Arm (AWS Graviton)。

Note

Graviton Arm 執行個體需要在 Linux 作業系統上建置 Amazon GameLift Servers 伺服器。C++ 和 C# 需要伺服器 SDK 5.1.1 或更新版本。Go 需要伺服器 SDK 5.0 或更新版本。這些執行個體在 out-of-the-box 可用支援。AL2023AL2

- 遊戲伺服器建置的運算、記憶體和儲存需求。
- 您計劃每個執行個體執行的伺服器程序數目。

透過使用較大的執行個體類型，您可以在每個執行個體上執行多個伺服器程序。這可以減少滿足玩家需求所需的執行個體數量。

如需詳細資訊：

- 關於執行個體類型，請參閱 [Amazon EC2 執行個體類型](#)。
- 關於每個執行個體執行多個程序，請參閱 [管理如何 Amazon GameLift Servers 啟動遊戲伺服器](#)。

Service Quotas

若要查看 的預設服務配額Amazon GameLift Servers，以及 的目前配額 AWS 帳戶，請執行下列動作：

- 如需 的一般服務配額資訊Amazon GameLift Servers，請參閱 中的[Amazon GameLift Servers端點和配額](#)AWS 一般參考。
- 如需您帳戶每個位置的可用執行個體類型清單，請開啟 Amazon GameLift Servers主控台的服務[配額](#)頁面。此頁面也會顯示您帳戶在每個位置中每個執行個體類型的目前用量。
- 如需每個區域執行個體類型的帳戶目前配額清單，請執行 AWS Command Line Interface (AWS CLI) 命令 [describe-ec2-instance-limits](#)。此命令會傳回您在預設區域中（或您指定的另一個區域中）擁有的作用中執行個體數量。

當您準備啟動遊戲時，請在 [Amazon GameLift Servers主控台](#) 中填寫啟動問卷。Amazon GameLift Servers 團隊使用啟動問卷來判斷遊戲的正確配額和限制。

管理 如何Amazon GameLift Servers啟動遊戲伺服器

您可以設定受管 EC2 機群的執行時間組態，以執行每個執行個體的多個遊戲伺服器程序。這會更有效率地使用您的託管資源。

機群如何管理多個程序

Amazon GameLift Servers 使用機群的執行時間組態來決定在每個執行個體上執行的程序類型和數量。執行時間組態包含至少一個代表一個遊戲伺服器可執行檔的伺服器程序組態。您可以定義其他伺服器程序組態，以執行與遊戲相關的其他程序類型。每個伺服器程序組態都會包含以下資訊：

- 遊戲建置中可執行檔的檔案名稱和路徑。
- (選用) 啟動時要傳遞至程序的參數。
- 要同時執行的程序數目。

當機群中的執行個體啟用時，它會啟動執行時間組態中定義的一組伺服器程序。使用多個程序時，Amazon GameLift Servers 會交錯啟動每個程序。伺服器程序的生命週期有限。隨著程序結束時，會 Amazon GameLift Servers啟動新的程序，以維護執行時間組態中定義的伺服器程序數量和類型。

您可以透過新增、變更或移除伺服器程序組態，隨時變更機群的執行時間組態。每個執行個體都會定期檢查機群執行時間組態的更新，以實作變更。以下是 如何Amazon GameLift Servers採用執行時間組態變更的方式：

1. 執行個體會將執行時間組態Amazon GameLift Servers最新版本的請求傳送至。
2. 執行個體會比較其作用中程序與最新的執行時間組態，然後執行下列動作：
 - 如果更新的執行時間組態移除伺服器程序類型，則此類型的作用中伺服器程序會繼續執行，直到結束為止。執行個體不會取代這些伺服器程序。
 - 如果更新的執行時間組態減少了伺服器程序類型的並行程序數量，則此類型的多餘伺服器程序會繼續執行，直到結束為止。執行個體不會取代這些多餘的伺服器程序。
 - 如果更新的執行時間組態新增新的伺服器程序類型，或增加現有類型的並行程序，則執行個體會啟動新的伺服器程序，直到達到Amazon GameLift Servers上限。在此情況下，執行個體會在現有程序結束時啟動新的伺服器程序。

針對多個程序最佳化機群

若要在機群上使用多個程序，請執行下列動作：

- [建立包含您要部署至機群之遊戲伺服器可執行檔的組建](#)，然後將組建上傳至 Amazon GameLift Servers。組建中的所有遊戲伺服器都必須在相同的平台上執行，並使用適用於 的伺服器 SDKAmazon GameLift Servers。
- 使用一或多個伺服器程序組態建立執行時間組態和多個並行程序。
- 將遊戲用戶端與 AWS SDK 版本 2016-08-04 或更新版本整合。

若要最佳化機群效能，建議您執行下列動作：

- 處理伺服器程序關閉案例，讓 Amazon GameLift Servers可以有效率地回收程序。例如：
 - 將關機程序新增至呼叫伺服器 API 的遊戲伺服器程式碼ProcessEnding()。
 - 在遊戲伺服器程式碼OnProcessTerminate()中實作回呼函數，以處理來自 的終止請求Amazon GameLift Servers。
- 請確定 Amazon GameLift Servers 關閉並重新啟動運作狀態不佳的伺服器程序。在遊戲伺服器程式碼中實作回OnHealthCheck()呼函數Amazon GameLift Servers，以將運作狀態回報給。Amazon GameLift Servers會自動關閉連續三個報告回報運作狀態不佳的伺服器程序。如果您不實作OnHealthCheck()，則 會Amazon GameLift Servers假設伺服器程序運作狀態良好，除非程序無法回應通訊。

選擇每個執行個體的程序數量

決定在執行個體上執行的並行程序數量時，請記住下列事項：

- Amazon GameLift Servers 會將每個執行個體限制為[並行程序的上限數](#)。機群伺服器程序組態的所有並行程序總和不得超過此配額。
- 為了維持可接受的效能層級，Amazon EC2 執行個體類型可能會限制可同時執行的程序數量。測試遊戲的不同組態，以尋找適合您偏好執行個體類型的正確程序數量。
- Amazon GameLift Servers 執行的並行程序不會超過設定的總數。這表示從先前的執行時間組態轉換到新組態可能會逐漸發生。

更新Amazon GameLift Servers機群組態

使用 Amazon GameLift Servers 主控台或 AWS CLI 來更新您的機群設定、變更遠端位置或刪除機群。對於受管機群，您無法變更機群的遊戲伺服器建置或執行個體類型。您必須取代機群。

Fast Build Update Tool (僅用於開發)

使用受管 EC2 機群，若要部署遊戲伺服器建置更新，您需要將每個新建置上傳至 [Amazon GameLift Servers](#) 並為其建立新的機群。

Fast Build Update Tool 可讓您在開發期間略過這些步驟，節省時間並加快開發反覆運算速度。使用此工具，您可以快速更新現有機群中所有運算的遊戲建置檔案。此工具有數個選項；您可以取代整個遊戲組建或變更 6 個特定檔案，也可以管理如何在更新後重新啟動遊戲伺服器程序。您也可以使用它來更新機群中的個別運算。

若要取得快速建置更新工具並進一步了解如何使用，請造訪 Github 中快速[建置更新工具](#) Amazon GameLift Servers 的工具組儲存庫。

您可以使用 Amazon GameLift Servers 主控台或 AWS CLI 更新可變機群屬性、連接埠設定和執行時間組態。若要變更擴展限制，請參閱 [使用 Amazon GameLift Servers 自動調整機群容量的規模](#)。

Amazon GameLift 伺服器 console

1. 在 [Amazon GameLift Servers 主控台](#) 的導覽窗格中，選擇機群。
2. 選擇您要更新的機群。機群必須處於 ACTIVE 狀態，才能編輯它。
3. 在機群詳細資訊頁面的下列任一區段中，選擇編輯。
 - 機群設定
 - 變更機群屬性，如 Name (名稱) 和 Description (說明) 等。
 - 新增或移除 Amazon CloudWatch 用來追蹤多個機群彙總指標的指標群組。Amazon CloudWatch Amazon GameLift Servers

- 更新資源建立限制設定。
- 開啟或關閉遊戲工作階段的防護。
- 執行期組態 – 您可以變更執行期組態的任何下列設定，並新增或移除執行期組態。
 - 變更遊戲伺服器的啟動路徑。
 - 新增、移除或變更選用的啟動參數。
 - 變更遊戲伺服器執行的並行程序數目。
- 遊戲工作階段啟用 – 透過更新最大並行遊戲工作階段啟用和新的啟用逾時，變更您希望伺服器程序執行和託管遊戲工作階段的方式。
- EC2 連接埠設定 – 更新允許機群傳入存取的 IP 地址和連接埠範圍。

4. 選擇確認以儲存變更。

AWS CLI

使用下列 AWS CLI 命令來更新機群：

- [update-fleet-attributes](#)
- [update-fleet-port-settings](#)
- [update-runtime-configuration](#)

更新機群位置

您可以使用Amazon GameLift Servers主控台或 CLI 新增或移除機群的 AWS 遠端位置。您無法變更機群的主區域。

Amazon GameLift 伺服器 console

1. 在[Amazon GameLift Servers主控台](#)的導覽窗格中，選擇機群。
2. 選擇您要更新的機群。機群必須處於 ACTIVE 狀態，才能編輯它。
3. 在機群詳細資訊頁面上，選擇位置索引標籤以檢視機群的位置。
4. 若要新增新的遠端位置，請選擇新增，然後選取您要部署執行個體的位置。此清單不包含機群執行個體類型無法使用的執行個體。
5. 選取新位置後，選擇新增。會將新位置Amazon GameLift Servers新增至清單，並將狀態設為 NEW。Amazon GameLift Servers然後開始在每個新增的位置佈建執行個體，並準備託管遊戲工作階段。

- 若要從機群中移除現有的遠端位置，請使用核取方塊來選取一或多個列出的位置。
- 選取一或多個機群後，選擇移除。移除的位置會保留在清單中，狀態設為 DELETING。Amazon GameLift Servers 然後開始在移除的位置中終止活動的程序。如果有作用中的執行個體託管遊戲工作階段，Amazon GameLift Servers 會使用遊戲伺服器終止程序來正常結束遊戲工作階段、終止遊戲伺服器，以及關閉執行個體。

AWS CLI

使用下列 AWS CLI 命令來更新機群位置：

- [create-fleet-locations](#)
- [delete-fleet-locations](#)

刪除機群

您可以刪除不再需要的機群。刪除機群會永久移除與遊戲工作階段和玩家工作階段相關聯的所有資料，以及收集的指標資料。或者您也可以保留叢集，停用自動調整規模，並手動將叢集調整為 0 個執行個體。

Note

如果機群具有 VPC 對等互連，請先呼叫 [CreateVpcPeeringAuthorization](#) 來請求授權。會在機群刪除期間 Amazon GameLift Servers 刪除 VPC 對等互連。

您可以使用 Amazon GameLift Servers 主控台或 CLI AWS 工具來刪除機群。

Amazon GameLift 伺服器 console

- 在 [Amazon GameLift Servers 主控台](#) 的導覽窗格中，選擇機群。
- 選擇您要刪除的機群。您只能刪除處於 ACTIVE 或 ERROR 狀態的機群。
- 選擇 刪除。
- 在刪除機群對話方塊中，輸入 以確認刪除 **delete**。
- 選擇 刪除。

AWS CLI

使用下列 AWS CLI 命令來刪除機群：

- [delete-fleet](#)

偵錯 Amazon GameLift Servers 機群問題

本主題提供如何解決Amazon GameLift Servers受管 EC2 機群問題的指引。

機群建立問題

當您建立受管 EC2 機群時，Amazon GameLift Servers服務會啟動建立機群的工作流程、在已安裝遊戲伺服器建置的情況下部署 EC2 執行個體，以及在每個執行個體上啟動遊戲伺服器程序。如需詳細說明，請參閱 [Amazon GameLift Servers 機群建立的運作方式](#)。機群在達到作用中狀態之前，無法託管遊戲工作階段和玩家。

您可以透過識別發生問題的機群建立階段並檢閱機群建立事件和日誌，來偵錯防止機群變為作用中的問題。如果日誌不提供有用的資訊，問題可能是因為內部服務錯誤。在這種情況下，請嘗試再次建立機群。如果問題仍然存在，請嘗試重新上傳遊戲組建來解決可能的檔案損毀）。您也可以聯絡 Amazon GameLift Servers 支援或在論壇上張貼問題。

下載並驗證組建

在此階段，會Amazon GameLift Servers取得您上傳的遊戲伺服器建置、擷取檔案，並執行任何安裝指令碼。如果機群建立在這些階段失敗，請查看機群事件和日誌以找出問題。可能的原因包括：

- Amazon GameLift Servers 無法取得壓縮的建置檔案（事件 FLEET_BINARY_DOWNLOAD_FAILED）。確認可以存取組建的儲存位置、您在與組建 AWS 區域相同的中建立機群，以及Amazon GameLift Servers具有正確存取許可的。
- Amazon GameLift Servers 無法擷取建置檔案（事件 FLEET_CREATION_EXTRACTING_BUILD）。
- 建置檔案中的安裝指令碼無法成功完成（事件 FLEET_CREATION_FAILED_INSTALLER）。

建置機群資源

此階段的問題通常涉及機群資源的配置和部署。可能的原因包括：

- 請求的執行個體類型無法使用。
- 請求的機群類型 (Spot 或隨需) 無法使用。

啟用遊戲伺服器程序

在此階段，Amazon GameLift Servers 正在嘗試許多任務並測試關鍵元素，包括遊戲伺服器的可行性、執行時間組態設定，以及遊戲伺服器使用 伺服器開發套件與 Amazon GameLift Servers 服務連線的能力。

Note

在此階段中，您可以遠端存取機群執行個體，以進一步調查問題。請參閱 [遠端連線至 Amazon GameLift Servers 機群執行個體](#)。

可能的問題包括：

- 伺服器程序不會開始執行。這表示機群的執行時間組態設定有問題（事件 FLEET_VALIDATION_LAUNCH_PATH_NOT_FOUND 或 FLEET_VALIDATION_EXECUTABLE_RUNTIME_FAILURE。確認您已正確設定啟動路徑和選用的啟動參數。
- 伺服器程序開始執行，但機群無法啟用。如果伺服器程序啟動並成功執行，但機群未移至作用中狀態，則可能原因是伺服器程序無法與服務通訊 Amazon GameLift Servers。確認您的遊戲伺服器正在進行這些正確的伺服器 SDK 呼叫（請參閱 [初始化伺服器程序](#)）：
 - 伺服器程序無法初始化（事件 SERVER_PROCESS_SDK_INITIALIZATION_TIMEOUT）。伺服器程序未成功呼叫 InitSdk()。
 - 伺服器程序無法在準備好託管遊戲工作階段（事件）Amazon GameLift Servers 時通知 SERVER_PROCESS_PROCESS_READY_TIMEOUT。伺服器程序已初始化，但未 ProcessReady() 及時呼叫。
- VPC 互連連線請求失敗。針對使用 VPC 對等互連（請參閱 [使用新機群設定 VPC 對等](#)）建立的叢集，VPC 對等互連會在此啟用階段完成。若 VPC 對等互連因為任何理由失敗，則新的叢集將無法移動至作用中狀態。您可以透過呼叫 [describe-vpc-peering-connections](#) 來追蹤對等請求的成功或失敗。請務必檢查是否存在有效的 VPC 對等互連授權（[describe-vpc-peering-authorizations](#)），因為授權只會在 24 小時內有效。

伺服器程序問題

伺服器程序已啟動，但很快便失敗或報告運作狀態不佳。

除了您的遊戲組建發生問題外，若您嘗試在執行個體上同時執行太多伺服器程序，便可能產生此結果。並行程序的最佳數量取決於執行個體類型和您遊戲伺服器的資源需求。請嘗試減少同時程序的

數量 (在叢集的執行時間組態中設定)，查看效能是否改善。您可以使用 Amazon GameLift Servers 主控台 (編輯機群的容量配置設定) 或呼叫 AWS CLI 命令 [update-runtime-configuration](#) 來變更機群的執行期組態。

機群刪除問題

叢集因執行個體數量上限而無法終止。

此錯誤訊息指出要刪除的叢集由於仍有作用中的執行個體，因此不允許刪除。您必須先將叢集向下擴展至零個作用中執行個體。您可以透過手動將叢集的所需執行個體數設為 "0"，接著等待向下擴展生效，來執行此作業。請務必關閉自動調整規模，因為這會阻止手動設定。

VPC 動作未獲得授權。

此問題僅適用於您特別為 建立 VPC 互連連線的機群 (請參閱 [適用於 Amazon GameLift Servers 的 VPC 對等互連](#))。這種情況是因為刪除機群的程序也包括刪除機群的 VPC 和任何 VPC 對等互連。您必須先呼叫 Amazon GameLift Servers [CreateVpcPeeringAuthorization\(\)](#) 的服務 API 或使用 AWS CLI 命令 `create-vpc-peering-authorization` 來取得授權。一旦您擁有授權，您便可以刪除叢集。

Amazon GameLift ServersRealtime 機群問題

僵屍遊戲工作階段：他們可以順利啟動及執行遊戲，但永遠不會結束。

您可能會在以下案例中觀察到此問題：

- 叢集的 Realtime 伺服器並未收到指令碼更新。
- 叢集快速地到達容量上限，且並未在使用者活動 (例如新的遊戲工作階段請求) 減少時向下擴展。

這基本上都是無法在您的 Realtime 指令碼中成功呼叫 `processEnding` 的結果。雖然叢集已進入作用中狀態，且遊戲工作階段也已啟動，但卻沒有任何方法停止他們。因此，正在執行遊戲工作階段的 Realtime 伺服器便無法進行釋放以啟動新的項目，導致新的遊戲工作階段只能在新的 Realtime 伺服器開機時啟動。此外，Realtime 指令碼的更新不會影響已在執行的遊戲工作階段，只會影響其他項目。

若要避免發生這種情況，指令碼需要提供觸發 `processEnding` 呼叫的機制。如 [Amazon GameLift ServersRealtime 指令碼範例](#) 中所示，其中一種方式是編寫閒置工作階段逾時的程式，在其中設定在特定時間長度內，若沒有任何玩家進行連線，便讓指令碼結束目前的遊戲工作階段。

但是，若您已陷入此案例中的情況，有數種因應措施可協助讓您的 Realtime 伺服器脫離困境。技巧是觸發 Realtime 伺服器程序 - 或基礎機群執行個體 - 重新啟動。在此情況下，Amazon GameLift Servers 會自動為您關閉遊戲工作階段。釋放 Realtime 伺服器後，他們便可以使用醉心版本的 Realtime 指令碼啟動新的遊戲工作階段。

取決於此問題存在的普遍狀況，有幾種方法可達到此目的：

- 向下擴展整個叢集。此方法是最簡單的方法，但其影響範圍也相當大。將整個叢集向下擴展至零個執行個體，等待叢集完全向下擴展，然後再往回向上擴展。這會抹除所有現有的遊戲工作階段，讓您使用最近更新的 Realtime 指令碼重新啟動。
- 遠端存取執行個體並重新啟動程序。若您只有少量的程序需要修復，此為良好的選項。若您已登入執行個體 (例如為了進行結尾記錄日誌或偵錯)，這可能會是最快的方法。請參閱 [遠端連線至 Amazon GameLift Servers 機群執行個體](#)。

若您選擇不在您的 Realtime 指令碼中包含呼叫 `processEnding` 的方式，即使叢集進入作用中狀態且已啟動遊戲工作階段，仍會有數種棘手的狀況。首先，正在執行的遊戲工作階段不會結束。因此，執行該遊戲工作階段的伺服器程序永遠不會進行釋放，以啟動新的遊戲工作階段。其次，Realtime 伺服器不會收到任何指令碼更新。

遠端連線至 Amazon GameLift Servers 機群執行個體

您可以連線到作用中 Amazon GameLift Servers 受管 EC2 機群中的任何執行個體。遠端存取執行個體的常見原因包括：

- 對遊戲伺服器整合的問題進行故障診斷。
- 微調您的執行時間組態和其他機群特定的設定。
- 取得即時遊戲伺服器活動，例如日誌追蹤。
- 使用實際玩家流量執行基準測試工具。
- 調查遊戲工作階段或伺服器程序的特定問題。

連線至執行個體時，請考慮下列潛在問題：

- 您可以連線到作用中機群中的任何執行個體。一般而言，您無法連線至非作用中的機群，例如正在啟用或處於錯誤狀態的機群。(這些機群在短時間內的可用性可能有限。)如需機群啟用問題的協助，請參閱 [偵錯 Amazon GameLift Servers 機群問題](#)。
- 連線至作用中執行個體不會影響執行個體的託管活動。執行個體會根據執行時間組態繼續啟動和停止伺服器程序。它會啟用並執行遊戲工作階段。執行個體可能會關閉以回應縮減事件或其他事件。

- 您對執行個體上的檔案或設定所做的任何變更都可能會影響執行個體的作用中遊戲工作階段和連線的玩家。

下列指示說明如何使用命令列界面 (CLI) AWS 遠端連線至執行個體。您也可以使用 AWS SDK 進行程式設計呼叫，如 [的服務 API 參考Amazon GameLift Servers](#)所述。

收集執行個體資料

若要連線至Amazon GameLift Servers受管 EC2 機群執行個體，您需要以下資訊：

- 您要連線的執行個體 ID。您可以使用執行個體 ID 或 ARN。
- 執行個體上使用之Amazon GameLift Servers版本的 伺服器 SDK。伺服器 SDK 已與執行個體上執行的遊戲組建整合。

下列指示說明使用 CLI AWS 完成這些任務的方式。您必須知道要連線之執行個體的機群 ID。

1. 取得運算名稱。取得機群中所有作用中運算的清單。使用機群 ID 或 ARN 呼叫 [list-compute](#)。對於單一位置機群，請僅指定機群識別符。對於多位置機群，指定機群識別符和位置。使用受管 EC2 機群時，會list-compute傳回機群執行個體的清單，而 屬性ComputeName是執行個體 ID。尋找您要存取的運算。

請求

```
aws gamelift list-compute \
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \
  --location ""sa-east-1"
```

回應

```
{
  "ComputeList": [
    {
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "FleetArn": "arn:aws:gamelift:us-west-2::fleet/fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "ComputeName": "i-0abc12d3e45fa6b78",
      "IpAddress": "00.00.000.00",
      "DnsName":
        "b08444ki909kvqu6zpw3is24x5pyz4b6m05i3jbxvpk9craztu0lqrbbrbnbkks.uwp57060n1k6dnlnw49b78hg1west-2.amazongamelift.com",
    }
  ]
}
```

```
    "ComputeStatus": "Active",
    "Location": "sa-east-1",
    "CreationTime": "2023-07-09T22:51:45.931000-07:00",
    "OperatingSystem": "AMAZON_LINUX_2023",
    "Type": "c4.large"
  }
]
```

2. 尋找伺服器 SDK 版本。如需此資訊，您需要查詢部署到機群的建置。伺服器開發套件版本是建置屬性。

- a. 使用機群 ID 或 ARN 呼叫 [describe-fleet-attributes](#)，以取得機群的建置 ID 和 ARN。
- b. 使用建置 ID 或 ARN 呼叫 [describe-build](#)，以取得建置的伺服器 SDK 版本。

例如：

請求

```
aws gamelift describe-fleet-attributes /
--fleet-ids "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

回應

```
{
  "FleetAttributes": [
    {
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "ComputeType": "EC2",
      "BuildId": "build-3333cccc-44dd-55ee-66ff-00001111aa22",
      . . .
    }
  ]
}
```

請求

```
aws gamelift describe-build /
--build-id "build-3333cccc-44dd-55ee-66ff-00001111aa22"
```


回應

```
"Build": {
  "BuildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "Name": "My_Game_Server_Build_One",
  "OperatingSystem": "AMAZON_LINUX_2023",

  "ServerSdkVersion": "5.1.1",
  . . .
}
```

連線至執行個體（伺服器 SDK 5)

如果您要連線的執行個體正在執行具有伺服器 SDK 5.x 版的遊戲組建，請使用 Amazon EC2 Systems Manager (SSM) 連線至執行個體。您可存取執行 Windows 或 Linux 的遠端執行個體。

 開始之前：

完成 SSM 設定步驟，並在本機電腦上安裝 SSM 外掛程式。如需詳細資訊，請參閱《Amazon EC2 Systems Manager 使用者指南》中的[設定 SSM](#) 和安裝 CLI 的 Session Manager 外掛程式。[AWS Amazon EC2 Systems Manager](#)

1. 要求存取執行個體登入資料。使用機群 ID 和您要連線之執行個體的運算名稱來呼叫 [get-compute-access](#)。會 Amazon GameLift Servers 傳回一組臨時登入資料來存取執行個體。例如：

請求

```
aws gamelift get-compute-access \
--compute-name i-11111111a222b333c \
--fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa
--region us-west-2
```

回應

```
{
  "ComputeName": " i-11111111a222b333c ",
  "Credentials": {
    "AccessKeyId": " ASIAIOSFODNN7EXAMPLE ",
```

```

    "SecretAccessKey": " wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY ",
    "SessionToken": " AQoDYXdzEJr...<remainder of session token>"
  },
  "FleetArn": " arn:aws:gamelift:us-west-2::fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa ",
  "FleetId": " fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa "
}

```

2. 匯出存取登入資料（選用）。您可以將登入資料匯出至環境變數，並使用它們來設定預設使用者的 AWS CLI。如需詳細資訊，請參閱 AWS Command Line Interface 《使用者指南》中的[設定 AWS CLI 的環境變數](#)。

```

export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>

```

3. 連線至機群執行個體。使用您要連線的執行個體啟動 SSM 工作階段。包含執行個體 AWS 的區域或位置。如需詳細資訊，包括如何設定 SSM 和 SSM 外掛程式，請參閱《Amazon EC2 Systems Manager 使用者指南》中的[啟動工作階段 \(AWS CLI\)](#)。

開始工作階段請求會自動使用您在步驟 1 中取得的登入資料。

```

aws ssm start-session \
--target i-11111111a222b333c \
--region us-west-2 \

```

Note

如果您收到存取遭拒的錯誤，您可能將 AWS_PROFILE 環境變數設定為 AWS 設定檔，這會導致 AWS CLI 使用錯誤的登入資料進行遠端存取。若要解決此問題，請暫時取消設定您的 AWS_PROFILE 環境變數。或者，您可以為您的遠端存取登入資料建立自訂 AWS 設定檔，並將 --profile 命令列參數新增至您的 start-session 請求。

連線至執行個體（伺服器 SDK 4.x 或更早版本）

如果您想要連線的執行個體正在執行具有伺服器 SDK 第 4 版或更早版本的遊戲組建，請使用下列指示。您可以連線到執行 Windows 或 Linux 的執行個體。使用遠端桌面通訊協定 (RDP) 用戶端連線至 Windows 執行個體。使用 SSH 用戶端連線至 Linux 執行個體。

1. 要求存取執行個體登入資料。當您有執行個體 ID 時，請使用命令 [get-instance-access](#) 來請求存取憑證。如果成功，Amazon GameLift Servers 將傳回執行個體的作業系統、IP 地址及一組登入資料 (使用者名稱及私密金鑰)。登入資料格式需視執行個體作業系統而定。請使用下列指示擷取 RDP 或 SSH 的登入資料。
 - 對於 Windows 執行個體 – 若要連線至 Windows 執行個體，RDP 需要使用者名稱和密碼。get-instance-access 要求會以簡單字串傳回這些值，因此您可直接使用傳回值。登入資料範例：

```
"Credentials": {
  "Secret": "aA1bBB2cCCd3EEE",
  "UserName": "gl-user-remote"
}
```

- 對於 Linux 執行個體 – 若要連線至 Linux 執行個體，SSH 需要使用者名稱和私有金鑰。Amazon GameLift Servers 會發出 RSA 私有金鑰，並以單一字串傳回它們，其中換行字元 (\n) 表示換行。若要讓私有金鑰可供使用，請採取下列步驟：(1) 將字串轉換為 .pem 檔案，以及 (2) 設定新檔案的許可。傳回的登入資料範例：

```
"Credentials": {
  "Secret": "-----BEGIN RSA PRIVATE KEY-----
nEXAMPLEKEYKCAQEAY7WZhaDsrA1W3mR1QtvhwyORRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/
\nvBoU7jLxx92pNHoFnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7i05dSrvC7dQkW2duV5QuUdE0QW
\nZ/aNxMniGQE6XAgfwlnXVBwrerrQo+ZWQeqiUwwMkuEbLeJfLhMCvYURpUMSC1oehm449ilx9X1F
\nG50TCFe0zfl8dqqCP6GzbPaIjiU19xX/az0R9V+tpU0zEL+wmXnZt3/nHPQ5xvD20JH67km6SuPW
\noPzev/D8V+x4+bHthfSjR9Y7DvQfjFBVwHXigBdtZcU2/wei8D/HYwIDAQABAOIBAGZ1kaEvnrqu
\n/uler7vgIn5m7lN5LKw4hJLAIW6tUT/fzvtCHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/
ufGxbL1\nmb5qwmGUnEpJaZD6QSSs3kICLwWUYUiGfc0uiSbmJoap/
GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2\nbahyWyJNfjLe4M86yd2YK3V2CmK+X/
B0sShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/
tJWSD9\n81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA30zdXzMqexXVJ1TLZVEH0E7bh1Y9d801ozR
\noQs/FiZNAx2iijCWyv01pjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfq1
+lIp1\nYkriL0DbLXlvRAH+yHPRit2hH0jtUNZh4Axv+cpg09qbUI3+43eEy24B7G/Uh
+GTfbjsXs0xQx/x\np9otyVwc7hsQ5TA5PZb
+mvkJ50BEKzet9XcKw0NBYELGhnEPE7cCgYEA06Vgov6YHleHui9kHuws
\nayav0elc5zkxjF9nfHFJRry21R1trw2Vdpn+9g481URrpzWV0Eihvm+xTtmaZ1Sp//lkq75XDwnU
\nWA8gkn603QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Unl2ajLivWUt5pbBrKbUC
\nngYBjb0+0Zk0sCcpZ29sbzjYjpIddErySiYRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
\noMo7yykABY70zd5wQewBQ4AdSlWSX4nGDtsiFxiWi5sKuAAe0CbTosy1s8w8fxoJ5Tz1sdoxNeGs
\nArq6Wv/G16zQuAE9zK9vwwKBgF+09VI/1wJBirsDGz9whVwFFPrTkJNvJZzYt69qezx1sjgFKshy
\nWBhd4xHZtmCqpBP1AymEjr/T0lboxyARMXmNIOWIANXMGb4KGSyl1mzSVAoQ+fqR+cJ3d0dyP11j
\nnjbb0Ed/NY8fr1NDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0i0egLda
-----"
```

```
\nNWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvtW+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
\nVRkAKKKYeGjKpUfVTrW0YFjXkfcrR/V+QFL50ndHAKJXjW7a4ejJLncTzmZSpYzwApc=\n-----END
RSA PRIVATE KEY-----",
  "UserName": "gl-user-remote"
}
```

使用 AWS CLI 時，您可以將 `--query` 和 `--output` 參數納入您的 `get-instance-access` 請求，以自動產生 `.pem` 檔案。

在 `.pem` 檔案設定權限時，請執行下列命令：

```
$ chmod 400 MyPrivateKey.pem
```

2. 開啟連接埠用於遠端連線。您可以透過機 Amazon GameLift Servers 群組態中授權的任何連接埠存取機群中的執行個體。您可使用 [describe-fleet-port-settings](#) 命令檢視機群連接埠設定。

我們建議的最佳實務，是僅在需要時開啟連接埠用於遠端存取，並於完成時加以關閉。您無法在建立機群之後，但在啟用之前更新連接埠設定。如果您卡住，請在開啟連接埠設定的情況下重新建立機群。

使用命令 [update-fleet-port-settings](#) 新增連接埠設定用於遠端連線 (例如 22 用於 SSH 或 3389 用於 RDP)。至於 IP 範圍值，請針對計畫用於連線的裝置指定 IP 地址 (轉換為 CIDR 格式)。範例：

```
$ AWS gamelift update-fleet-port-settings
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
  --inbound-permission-authorizations
    "FromPort=22,ToPort=22,IpRange=54.186.139.221/32,Protocol=TCP"
```

下列範例會在 Windows 機群上開啟連接埠 3389

```
$ AWS gamelift update-fleet-port-settings
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
  --inbound-permission-authorizations
    "FromPort=3389,ToPort=3389,IpRange=54.186.139.221/32,Protocol=TCP"
```

3. 開啟遠端連線用戶端。Windows 執行個體請使用遠端桌面，Linux 執行個體請使用 SSH。使用 IP 地址、連接埠設定及存取登入資料連線至執行個體。

SSH 範例：

```
ssh -i MyPrivateKey.pem gl-user-remote@192.0.2.0
```

檢視遠端執行個體上的檔案

於遠端連線執行個體時，您享有完整的使用者及管理存取。這代表您也可能在遊戲主機造成錯誤及故障。如果執行個體託管具有作用中玩家的遊戲，則您會面臨當機遊戲工作階段和捨棄玩家，或中斷遊戲關閉程序，並導致儲存的遊戲資料和日誌發生錯誤的風險。

在託管執行個體上尋找這些資源：

- 遊戲建置檔案。這些檔案是您上傳至的遊戲組建Amazon GameLift Servers。其中包括一或多個遊戲伺服器可執行檔、資產和相依性。遊戲組建檔案位於名為的根目錄中game：
 - 在 Windows 上：c:\game
 - 在 Linux 上：/local/game
- 遊戲記錄檔。尋找遊戲伺服器在您指定的目錄路徑game根目錄中產生的日誌檔案。
- Amazon GameLift Servers 託管資源。根目錄Whitewater包含 Amazon GameLift Servers服務用來管理遊戲託管活動的檔案。請勿因任何原因修改這些檔案。
- 執行時間組態。請勿存取個別執行個體的執行時間組態。若要變更執行時間組態屬性，請更新機群的執行時間組態（請參閱 AWS SDK 操作 [UpdateRuntimeConfiguration](#) 或 AWS CLI [update-runtime-configuration](#)）。
- 機群資料。JSON 檔案包含執行個體所屬機群的相關資訊，供執行個體上執行的伺服器程序使用。JSON 檔案位於下列位置：
 - 在 Windows 上：C:\GameMetadata\gamelift-metadata.json
 - 在 Linux 上：/local/gamemetadata/gamelift-metadata.json
- TLS 憑證。如果執行個體位於已啟用 TLS 憑證產生功能的機群，請在下列位置尋找憑證檔案，包括憑證、憑證鏈、私有金鑰和根憑證：
 - 在 Windows 上：c:\GameMetadata\Certificates
 - 在 Linux 上：/local/gamemetadata/certificates/

適用於 Amazon GameLift Servers 的 VPC 對等互連

本主題提供如何在您的 Amazon GameLift Servers 託管遊戲伺服器和其他非 Amazon GameLift Servers 資源間設定 VPC 對等互連的指導。使用 Amazon Virtual Private Cloud (VPC) 對等連線，讓您的遊戲伺服器能夠直接和私下與您的其他 AWS 資源通訊，例如 Web 服務或儲存庫。您可以使用在上執行的任何資源建立 VPC 對等互連，AWS 並由您有權存取 AWS 的帳戶管理。

Note

VPC 對等互連是進階功能。若要了解讓遊戲伺服器能夠直接和私下與其他 AWS 資源通訊的偏好選項，請參閱 [與機群中的其他 AWS 資源通訊](#)。

如果您已經熟悉 Amazon VPCs 和 VPC 對等互連，請了解與 Amazon GameLift Servers 遊戲伺服器的對等互連設定有些不同。您無法存取包含遊戲伺服器的 VPC，而該 VPC 是由 Amazon GameLift Servers 服務所控制，因此您無法直接為其請求 VPC 對等互連。您會改為先授權 VPC 使用非 Amazon GameLift Servers 資源的權利，以接受來自 Amazon GameLift Servers 服務的對等互連請求。然後觸發 Amazon GameLift Servers 以請求您剛授權的 VPC 對等。Amazon GameLift Servers 會處理建立對等連線、設定路由表以及設定連線的任務。

為現有機群設定 VPC 對等互連

1. 取得 AWS 帳戶 ID 和（登入資料）。

您需要下列 AWS 帳戶的 ID 和登入憑證。您可以登入 [AWS Management Console](#) 並檢視您的帳戶設定，以尋找帳戶 AWS IDs。若要取得登入資料，請前往 IAM 主控台。

- AWS 用來管理 Amazon GameLift Servers 遊戲伺服器的 帳戶。
- AWS 用於管理非 Amazon GameLift Servers 資源的 帳戶。

如果您為 Amazon GameLift Servers 和非 Amazon GameLift Servers 資源使用相同的帳戶，則只需要該帳戶的 ID 和登入資料。

2. 取得每個 VPC 的識別符。

取得要進行對等互連兩個 VPC 的以下資訊：

- Amazon GameLift Servers 遊戲伺服器的 VPC – 這是您的 Amazon GameLift Servers 機群 ID。您的遊戲伺服器會在 EC2 執行個體叢集上的 Amazon GameLift Servers 內部署。叢集會自動置

放在其自身擁有的 VPC 內，且該 VPC 是由 Amazon GameLift Servers 服務進行管理。您無法直接存取 VPC，因此會以機群 ID 識別。

- 非 Amazon GameLift Servers AWS 資源的 VPC – 您可以使用在 上執行的任何資源建立 VPC 對等互連，AWS 並由您有權存取 AWS 的帳戶管理。如果您尚未為這些資源建立 VPC，請參閱 [Amazon VPC 入門](#)。建立 VPC 後，您可以透過登入 Amazon VPC 的 [AWS Management Console](#) 並檢視您的 VPC，來找到 VPC ID。

 Note

設定對等互連時，兩個 VPC 都必須存在相同區域。您 Amazon GameLift Servers 叢集遊戲伺服器的 VPC 位於和叢集相同的區域內。

3. 授權 VPC 對等。

在此步驟中，您會預先授權未來來自 Amazon GameLift Servers 的請求，允許其針對非 Amazon GameLift Servers 資源與 VPC 建立與遊戲伺服器之 VPC 的對等互連。此動作會更新 VPC 的安全群組。

若要授權 VPC 對等互連，請呼叫服務 API [CreateVpcPeeringAuthorization\(\)](#) 或使用 AWS CLI 命令 `create-vpc-peering-authorization`。使用管理您非 Amazon GameLift Servers 資源的帳戶進行此呼叫。請識別以下資訊：

- 對等 VPC ID – 這是針對具有非 Amazon GameLift Servers 資源的 VPC。
- Amazon GameLift Servers AWS 帳戶 ID – 這是您用來管理 Amazon GameLift Servers 機群的帳戶。

授權 VPC 對等互連後，授權會保持有效 24 小時，除非撤銷。您可以使用以下操作來管理 VPC 對等授權：

- [DescribeVpcPeeringAuthorizations\(\)](#) (AWS CLI `describe-vpc-peering-authorizations`)。
- [DeleteVpcPeeringAuthorization\(\)](#) (AWS CLI `delete-vpc-peering-authorization`)。

4. 要求對等連線。

透過有效的授權，您可以請求 Amazon GameLift Servers 建立對等連線。

若要請求 VPC 對等互連，請呼叫服務 API [CreateVpcPeeringConnection\(\)](#) 或使用 AWS CLI 命令 `create-vpc-peering-connection`。使用管理您 Amazon GameLift Servers 遊戲伺服器的帳戶進行此呼叫。使用下列資訊來識別您要建立對等互連的兩個 VPC：

- 對等 VPC ID 和 AWS 帳戶 ID – 這是非Amazon GameLift Servers 資源的 VPC，以及您用來管理它們的帳戶。VPC ID 必須與有效對等授權上的 ID 相符。
- 機群 ID – 這可識別Amazon GameLift Servers遊戲伺服器的 VPC。

5. 追蹤對等連線狀態。

要求 VPC 對等連線是一種非同步操作。若要追蹤對等請求的狀態並處理成功或失敗情況，請使用以下其中一個選項：

- 使用 `DescribeVpcPeeringConnections()` 持續輪詢。這個操作會擷取 VPC 對等連線記錄 (包括請求的狀態)。如果成功建立對等連線，連線記錄還包含指派給 VPC 之私有 IP 地址的 CIDR 區塊。
- 使用 [DescribeFleetEvents\(\)](#) 處理與 VPC 對等互連相關聯的叢集事件，包括成功和失敗事件。

建立對等連線之後，您可以使用下列操作來進行管理：

- [DescribeVpcPeeringConnections\(\)](#) (AWS CLI `describe-vpc-peering-connections`)。
- [DeleteVpcPeeringConnection\(\)](#) (AWS CLI `delete-vpc-peering-connection`)。

使用新機群設定 VPC 對等

您可以建立新的 Amazon GameLift Servers 機群，並同時請求 VPC 對等連線。

1. 取得 AWS 帳戶 ID 和 (登入資料)。

您需要以下兩個 AWS 帳戶的 ID 和登入憑證。您可以登入 [AWS Management Console](#) 並檢視您的帳戶設定，以尋找帳戶 AWS IDs。若要取得登入資料，請前往 IAM 主控台。

- AWS 用來管理Amazon GameLift Servers遊戲伺服器的 帳戶。
- AWS 用於管理非Amazon GameLift Servers 資源的 帳戶。

如果您為 Amazon GameLift Servers和非Amazon GameLift Servers 資源使用相同的帳戶，則只需要該帳戶的 ID 和登入資料。

2. 取得非 Amazon GameLift Servers AWS 資源的 VPC ID。

如果您尚未為這些資源建立 VPC，請立即執行此操作（請參閱 [Amazon VPC 入門](#)）。請確定您已在計劃建立新機群的相同區域中建立新 VPC。如果您的非 Amazon GameLift Servers 資源是以與搭配使用的不同 AWS 帳戶或使用者/使用者群組管理 Amazon GameLift Servers，則在下一個步驟中請求授權時，您將需要使用這些帳戶登入資料。

一旦建立了 VPC，您可以透過檢視 VPC，在 Amazon VPC 主控台中尋找 VPC ID。

3. 授權 VPC 對等互連使用非 Amazon GameLift Servers 資源的權利。

當 Amazon GameLift Servers 建立新機群及對應 VPC 時，它也會傳送請求，以和您非 Amazon GameLift Servers 資源的 VPC 建立對等互連。您需要預先授權該請求。此步驟會更新您 VPC 的安全群組。

使用管理非 Amazon GameLift Servers 資源的帳戶登入資料，呼叫服務 API [CreateVpcPeeringAuthorization\(\)](#) 或使用 AWS CLI 命令 `create-vpc-peering-authorization`。請識別以下資訊：

- 對等 VPC ID – VPC 與非 Amazon GameLift Servers 資源的 ID。
- Amazon GameLift Servers AWS 帳戶 ID – 您用來管理 Amazon GameLift Servers 機群的帳戶 ID。

授權 VPC 對等互連後，授權會保持有效 24 小時，除非撤銷。您可以使用以下操作來管理 VPC 對等授權：

- [DescribeVpcPeeringAuthorizations\(\)](#) (AWS CLI `describe-vpc-peering-authorizations`)。
- [DeleteVpcPeeringAuthorization\(\)](#) (AWS CLI `delete-vpc-peering-authorization`)。

4. 遵循 [使用 CLI AWS 建立新機群](#) 的指示。包含下列其他參數：

- `peer-vpc-aws-account-id` – 您用來使用非 Amazon GameLift Servers 資源管理 VPC 之帳戶的 ID。
- `peer-vpc-id` – VPC 與您非 Amazon GameLift Servers 帳戶的 ID。

使用 VPC 對等互連參數成功呼叫 [create-fleet](#) 會產生新的叢集及新的 VPC 對等互連請求。會將機群狀態設為新建且會起始化機群啟動程序。對等連線請求的狀態是設定為 `initiating-request`。您可以透過呼叫 [describe-vpc-peering-connections](#) 來追蹤對等請求的成功或失敗。

同時請求新機群和 VPC 對等連線時，兩個動作會一起成功或失敗。若叢集在建立程序期間失敗，便不會建立 VPC 對等互連。同樣地，若 VPC 對等互連因任何原因失敗，新叢集也將無法從正在啟用狀態移動到作用中狀態。

Note

在機群準備好成為作用中前，新 VPC 對等連線都未完成。這表示連線無法使用，且無法在遊戲伺服器建置安裝程序中使用。

下列範例會在預先建立的 VPC 和新機群的 VPC 之間建立新機群和對等連線。預先建立的 VPC 由非 Amazon GameLift Servers AWS 帳戶 ID 和 VPC ID 的組合唯一識別。

```
$ AWS gamelift create-fleet
  --name "My_Fleet_1"
  --description "The sample test fleet"
  --ec2-instance-type "c5.large"
  --fleet-type "ON_DEMAND"
  --build-id "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff"
  --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
                           MaxConcurrentGameSessionActivations=2,
                           ServerProcesses=[{LaunchPath=C:\game\Bin64.dedicated
\MultiplayerSampleProjectLauncher_Server.exe,
                                           Parameters="+sv_port 33435 +start_lobby,
                                           ConcurrentExecutions=10}]"
  --new-game-session-protection-policy "FullProtection"
  --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
                                   PolicyPeriodInMinutes=15"
  --ec2-inbound-permissions
  "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"

  "FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP"
  --metric-groups "EMEAfleets"
  --peer-vpc-aws-account-id "111122223333"
  --peer-vpc-id "vpc-a11a11a"
```

可複製的版本：

```
AWS gamelift create-fleet --name "My_Fleet_1" --description "The
sample test fleet" --fleet-type "ON_DEMAND" --metric-groups
"EMEAfleets" --build-id "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff"
--ec2-instance-type "c5.large" --runtime-configuration
```

```
"GameSessionActivationTimeoutSeconds=300,MaxConcurrentGameSessionActivations=2,ServerProcesses=
\game\Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe,Parameters=
+sv_port 33435 +start_lobby,ConcurrentExecutions=10}]]" --new-game-session-
protection-policy "FullProtection" --resource-creation-limit-policy
"NewGameSessionsPerCreator=3,PolicyPeriodInMinutes=15" --ec2-inbound-
permissions "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
"FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP" --peer-vpc-aws-account-id
"111122223333" --peer-vpc-id "vpc-a11a11a"
```

對 VPC 對等問題進行疑難排解

如果您無法為 Amazon GameLift Servers 遊戲伺服器建立 VPC 對等連線，請考慮下列常見的根本原因：

- 找不到所請求連線的授權：
 - 檢查非 Amazon GameLift Servers VPC 的 VPC 授權狀態。它可能不存在或可能已過期。
 - 檢查您嘗試建立對等互連的兩個 VPC 區域。如果它們不在同一個區域，則無法建立其對等互連。
- 兩個 [VPC 的 CIDR 區塊](#) ([請參閱無效的 VPC 對等互連組態](#)) 重疊。VPCs 指派給對等 VPC 的 IPv4 CIDR 區塊不能重疊。機 Amazon GameLift Servers 群 VPC 的 CIDR 區塊會自動指派且無法變更，因此您需要為非 Amazon GameLift Servers 資源變更 VPC 的 CIDR 區塊。若要解決此問題：
 - 透過呼叫 `DescribeVpcPeeringConnections()` 來查詢 Amazon GameLift Servers 機群的 CIDR 區塊。
 - 前往 Amazon VPC 主控台，為非 Amazon GameLift Servers 資源尋找 VPC，然後變更 CIDR 區塊，使其不重疊。
- 新的機群未啟用 (在與新機群請求 VPC 對等時)。如果新機群無法進入作用中狀態，表示沒有可以建立對等互連的 VPC，因此對等連線無法成功。

Amazon GameLift Servers 受管容器機群

Amazon GameLift Servers 受管容器機群提供以雲端為基礎的資源來託管您的容器化遊戲伺服器軟體。透過受管容器機群，您可以取得 AWS 雲端資源的彈性、安全性和可靠性，並針對多玩家遊戲託管進行最佳化。Amazon GameLift Servers 提供強大的主機管理工具。

 使用這些受管容器的工具加速入門：

- [容器入門套件](#)可簡化整合和機群設定。它會為您的遊戲伺服器新增必要的遊戲工作階段管理功能，並使用預先設定的範本，為您的遊戲伺服器建置容器機群和自動化部署管道。部署之

後，請使用 Amazon GameLift Servers 主控台和 API 工具來監控機群效能、管理遊戲工作階段和分析指標。

- 對於 Unreal Engine 或 Unity 開發人員，請使用遊戲引擎 [Amazon GameLift Servers 外掛程式和伺服器 SDKs](#) 來整合遊戲伺服器，並從遊戲引擎的開發環境內部建置容器機群。外掛程式的引導式工作流程可協助您建立快速、簡單的解決方案，並使用受管容器進行雲端託管。您可以在此基礎上建置，為您的遊戲建立自訂託管解決方案。

受管容器機群是一組執行 Linux 的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，其 Amazon GameLift Servers 擁有並根據您的組態運作。這些執行個體實際位於支援的 AWS 區域或 Local Zones。建立容器機群時，您可以選擇符合遊戲伺服器對運算能力、記憶體、儲存體、聯網功能需求的 EC2 執行個體類型。

對於受管容器機群，您可以將 Linux 型容器映像存放在 Amazon Elastic Container Registry (Amazon ECR) 儲存庫中，並建立容器群組定義來描述容器架構。當您建立機群時，會使用最新版本的 Linux Amazon Machine Image (AMI) Amazon GameLift Servers 佈建機群執行個體，並使用容器群組定義來部署容器映像。即使您更新容器群組定義或變更容器映像，容器機群中的所有執行個體也會使用相同的 AMI 版本。

Note

根據最佳實務，建議您每 30 天替換一次機群，以維護託管遊戲伺服器的安全 up-to-date 執行期環境。這需要建立新的機群，並將玩家流量遷移到其中。如需詳細的指導方針，請參閱 [Amazon GameLift Servers 的安全最佳實務](#)。

部署容器化執行個體之後，容器會開始啟動遊戲伺服器程序。每個遊戲伺服器程序都會建立 Amazon GameLift Servers 服務的連線、報告託管遊戲工作階段的準備程度，並開始通訊運作狀態。然後 Amazon GameLift Servers 可以提示伺服器程序啟動遊戲工作階段。

除了機群部署之外，還會 Amazon GameLift Servers 處理下列主機管理任務，因此您不需要：

- 追蹤機群中所有容器的狀態，並取代過時或運作狀態不佳的容器。
- 處理伺服器程序與服務之間通訊的身分驗證 Amazon GameLift Servers。
- 提供自動調整規模工具，可動態調整機群容量以滿足玩家需求。
- 報告機群 EC2 執行個體、容器和伺服器程序的效能指標。

請參閱這些主題，了解如何設定和維護受管容器機群：

- [使用Amazon GameLift Servers受管容器託管的開發藍圖](#)
- [建立Amazon GameLift Servers受管容器機群](#)
- [自訂Amazon GameLift Servers容器機群](#)
- [使用 擴展遊戲託管容量 Amazon GameLift Servers](#)
- [更新Amazon GameLift Servers受管容器機群](#)

建立Amazon GameLift Servers受管容器機群

建立Amazon GameLift Servers受管容器機群，以在 AWS 雲端中部署和託管容器化遊戲伺服器。當您建立容器機群時，請指定容器群組定義，以指定一或多個容器映像（至少包含您的遊戲伺服器組建）和組態設定。

當您建立新的受管容器機群資源時，您會立即啟動機群建立的第一階段。當 Amazon GameLift Servers 佈建 EC2 執行個體、安裝執行時間環境、將容器群組部署到執行個體，並開始啟動遊戲伺服器程序時，受管機群建立會經過數個階段。根據遊戲伺服器建置所需的執行時間環境，Amazon GameLift Servers 會在機群建立時部署最新版本的 Amazon Machine Image (AMI)（且機群中的所有未來執行個體將使用相同的版本）。您可以在主控台或使用 AWS Command Line Interface () 監控機群的狀態 AWS CLI。當機群準備好託管遊戲工作階段時，狀態會變更為 ACTIVE。如需建立叢集問題的協助，請參閱[偵錯 Amazon GameLift Servers 機群問題](#)。

您可以選擇建立空的容器機群，然後在稍後新增或更新機群的容器群組定義。如果您建立沒有容器群組定義的機群，機群將不會達到作用中狀態。

Note

根據最佳實務，我們建議您每 30 天替換一次機群，以維護託管遊戲伺服器的安全up-to-date 執行期環境。這需要建立新的機群，並將玩家流量遷移到其中。如需詳細的指導方針，請參閱[Amazon GameLift Servers 的安全最佳實務](#)。

使用 [Amazon GameLift Servers主控台](#) 或 AWS Command Line Interface (AWS CLI) 來建立容器機群。

Console

在 [Amazon GameLift Servers 主控台](#) 中，選取您要建立機群的 AWS 區域。容器群組定義必須位於您要建立機群的相同區域。

開啟主控台的左側導覽列，然後選擇受管容器：機群。在機群頁面上，選擇建立容器機群。

步驟 1：定義受管容器機群詳細資訊

1. 在容器機群詳細資訊區段中，輸入機群描述。
2. 指定機群的 IAM 角色。此角色具有 Amazon GameLift Servers 必須代表您管理容器機群的許可。如需建立所需服務角色的說明，請參閱 [設定的 IAM 服務角色 Amazon GameLift Servers](#)。
3. 選擇日誌組態選項。預設會選取 CloudWatch 選項。根據您選取的選項提供必要資訊。
4. 將容器群組新增至機群。此為選用步驟。您可以選擇建立沒有容器群組的機群，並計劃稍後新增。沒有任何容器群組的機群將不會部署任何機群執行個體，且尚無法託管任何遊戲，但會建立機群資源。
 - 選取遊戲伺服器容器群組定義。選擇性地指定您要部署的定義版本。如果您未指定版本編號，Amazon GameLift Servers 會自動使用最新版本。
 - 選擇性地新增每個執行個體容器群組定義和版本。如果您未指定版本編號，Amazon GameLift Servers 會自動使用最新版本。
5. 在其他詳細資訊中，您可以設定一些選用的自訂項目。建立容器機群不需要這些設定。

步驟 2：定義執行個體詳細資訊

1. 在執行個體部署中，選取要部署執行個體的一或多個遠端位置。系統會自動選取主要區域（這是要建立機群的區域）。如果您選擇其他位置，機群執行個體也會部署在這些位置。

Important

若要使用預設未啟用的區域，請在您的 中啟用它們 AWS 帳戶。

- 在 2022 年 2 月 28 日之前建立的未啟用區域的機群不受影響。
- 若要建立新的多位置機群或更新現有的多位置機群，請先啟用您選擇使用的任何區域。

如需未預設啟用的區域以及如何啟用它們的詳細資訊，請參閱《》中的[管理 AWS 區域](#) AWS 一般參考。

2. 選取機群的執行個體組態。主控台會自動計算所需的最低 vCPU 和記憶體（根據您為每個容器群組設定的總限制）。它會根據資源需求和您輸入的位置，篩選可用執行個體類型的完整清單。您可以視需要新增其他篩選條件。

如需選擇執行個體類型的詳細資訊，請參閱[設定容器機群](#)。您選擇的執行個體類型大小會影響遊戲伺服器容器群組封裝到每個機群執行個體的方式。根據您的選擇，請考慮檢閱每個執行個體所需遊戲伺服器容器群組的設定。

步驟 4：檢閱和建立

- 檢閱您的機群組態設定。

無論機群狀態為何，您隨時都可以更新機群的中繼資料和組態。如需詳細資訊，請參閱[更新 Amazon GameLift Servers 機群組態](#)。您可以在機群達到 ACTIVE 狀態後更新機群容量。如需詳細資訊，請參閱[使用 擴展遊戲託管容量 Amazon GameLift Servers](#)。您也可以新增或移除遠端位置。

檢閱完成後，請選擇建立。

如果您的請求成功，主控台會顯示新機群資源的詳細資訊頁面。一開始狀態為 NEW，因為會 Amazon GameLift Servers 開始機群建立程序。您可以在 Fleets (叢集) 頁面追蹤新叢集的狀態。當機群達到 狀態時，已準備好託管遊戲工作階段 ACTIVE。

AWS CLI

若要使用 建立容器機群 AWS CLI，請開啟命令列視窗並使用 `create-container-fleet` 命令。如需此命令的詳細資訊，請參閱《AWS CLI 命令參考》[create-container-fleet](#) 中的。

以下顯示的範例 `create-container-fleet` 請求會建立具有下列特性的新容器機群：

- ContainerGroupsConfiguration 僅指定遊戲伺服器容器群組定義：MyAdventureGameContainerGroup。將部署到每個機群執行個體的遊戲伺服器容器群組數量由 計算 Amazon GameLift Servers。
- 根據預設，機群會使用 `c5.large` 隨需執行個體。

- 根據預設，機群會開啟一組連線連接埠和傳入許可連接埠，由 計算Amazon GameLift Servers。它會將容器群組部署至下列位置：

```
aws gamelift create-container-fleet \  
  --fleet-role-arn arn:aws:iam::MyAccount:role/MyContainersRole \  
  --game-server-container-group-definition-name "rn:aws:gamelift:us-  
west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2" \  

```

如果建立機群請求成功，會Amazon GameLift Servers傳回一組機群屬性，其中包含您請求的組態設定和新的容器機群 ID。Amazon GameLift Servers 然後將機群狀態和位置狀態設定為新增，並啟動機群啟用程序。您可以追蹤機群的狀態，並使用這些 CLI 命令檢視其他機群資訊：

- [describe-fleet-events](#)
- [describe-container-fleet](#)
- [describe-fleet-capacity](#)
- [describe-fleet-port-settings](#)
- [describe-fleet-utilization](#)
- [describe-fleet-location-capacity](#)
- [describe-fleet-location-utilization](#)

您可以使用以下命令，視需要變更叢集的容量及其他組態設定：

- [update-container-fleet](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

更新Amazon GameLift Servers受管容器機群

您可以更新受管容器機群的大部分屬性，包括容器群組定義。根據要更新的設定，機群更新可能會啟動新的機群部署。在機群部署中，機群中的所有執行個體都會移除，並以具有新組態的執行個體取代。需要部署的設定包括：

- 容器群組定義，包括容器映像的更新
- 連線連接埠範圍和傳入許可
- 日誌組態

您可以在 [Amazon GameLift Servers 主控台](#) 或 AWS Command Line Interface (AWS CLI) 中追蹤機群部署的狀態，以建立容器機群。

Note

您無法更新機群的執行時間環境。在機群建立時，機群的 Amazon Machine Image (AMI) 會設定為 Linux AMI 的最新版本。部署到此機群的所有容器映像都必須與此版本相容。若要變更機群的 AMI 或升級至較新的版本，您必須建立新的機群。根據最佳實務，建議您每 30 天替換一次機群，以維護託管遊戲伺服器的安全 up-to-date 執行期環境。如需詳細的指導方針，請參閱 [Amazon GameLift Servers 的安全最佳實務](#)。

Console

在 [Amazon GameLift Servers 主控台](#) 中，選取您要建立機群的 AWS 區域。容器群組定義必須位於您要建立機群的相同區域。

開啟主控台的左側導覽列，然後選擇受管容器：機群。在受管容器機群頁面上，從清單中選擇機群，然後選擇編輯。

1. 視需要更新容器機群設定。完成時，請選擇 Create (建立)。
2. 如果您的更新需要機群部署，系統會要求您指定部署選項，如下所示：
 - 遊戲工作階段保護。您可以選擇保護具有作用中遊戲工作階段（安全部署）的機群執行個體。使用此設定，在遊戲工作階段結束之前，不會取代機群執行個體。或者，您可以選擇取代機群執行個體，無論遊戲工作階段活動為何（不安全的部署）。不安全的部署在開發和測試階段非常有用，以減少部署時間。
 - 運作狀態百分比下限。您可以管理機群執行個體的取代速度。使用此設定可在部署期間維持最少量的運作狀態任務。較低的值會優先考慮部署速度，而較高的值可確保遊戲伺服器在整個部署期間保持高可用性。
 - 部署失敗策略。決定部署失敗時要採取的動作。部署失敗表示某些更新的容器狀態檢查失敗，並被視為受損。您可以設定部署，將所有機群執行個體自動復原至先前部署的狀態。或者，您可以選擇維護一些受損的機群執行個體，以用於偵錯。

如果您的請求成功，主控台會顯示受管容器機群的部署索引標籤。使用此索引標籤來追蹤每個部署的狀態。如果您啟動機群的新部署，此動作會自動取消機群目前正在進行的任何部署。

AWS CLI

若要使用 建立容器機群 AWS CLI，請開啟命令列視窗並使用 `update-container-fleet` 命令。如需此命令的詳細資訊，請參閱《AWS CLI 命令參考[update-container-fleet](#)》中的 。

下列範例會更新具有下列特性的現有容器機群：

- 它會更新遊戲伺服器容器群組定義，以使用第 2 版。
- 它會指定安全部署選項。

```
{
  "DeploymentConfiguration": {
    "ImpairmentStrategy": "ROLLBACK",
    "MinimumHealthyPercentage": 75,
    "ProtectionStrategy": "WITH_PROTECTION"
  },
  "FleetId": "containerfleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "GameServerContainerGroupDefinitionName": "arn:aws:gamelift:us-west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2"
}
```

自訂Amazon GameLift Servers容器機群

本節中的主題說明Amazon GameLift Servers受管容器的一些選用功能。您可以選擇使用任何或所有這些功能。

主題

- [設定資源限制](#)
- [指定必要容器](#)
- [設定網路連線](#)
- [設定容器的運作狀態檢查](#)
- [設定容器相依性](#)
- [設定容器機群](#)

設定資源限制

對於每個容器群組，您可以判斷容器群組執行其軟體所需的記憶體和運算能力。Amazon GameLift Servers 依賴此資訊來管理整個容器群組的資源。它也會使用此資訊來計算機群映像可以容納多少遊戲伺服器容器群組。您也可以設定個別容器的限制。

您可以設定容器群組的記憶體和運算能力上限。根據預設，群組中的所有容器都會共用這些資源。您可以設定個別容器的限制，進一步自訂資源管理。

設定個別容器的選用限制

設定容器特定的資源限制可讓您對個別容器如何使用群組的資源進行更大的控制。如果您未設定容器特定的限制，群組中的所有容器都會共用群組資源。共用可提供更大的彈性，以在需要時使用資源。它也會提高程序彼此競爭並導致容器故障的可能性。

為任何容器設定下列任何ContainerDefinition屬性。

- `MemoryHardLimitMebibytes` – 設定容器的記憶體上限。如果容器超過此限制，則會導致重新啟動。
- `Vcpu limit` – 預留最低數量的 vCPU 資源，供容器專屬使用。容器一律有可用的預留數量。如果有其他資源可用，則可以隨時超過此最小值。(1024 CPU 單位相當於 1 個 vCPU。)

設定容器群組的總資源限制

如果您設定個別容器的限制，您可能需要修改容器群組所需的記憶體和 vCPU 資源數量。目標是配置足夠的資源來最佳化遊戲伺服器效能。Amazon GameLift Servers 使用這些限制來計算如何在機群執行個體上封裝遊戲伺服器容器群組。您也會在為容器機群選擇執行個體類型時使用它們。

計算容器群組所需的總記憶體和 vCPU。考慮下列各項：

- 容器群組中所有容器執行的所有程序有哪些？新增這些程序所需的資源。請記下任何容器特定的限制。
- 您計劃在每個容器群組中執行多少個並行遊戲伺服器程序？您可以在遊戲伺服器容器映像中確定這一點。

根據您對容器群組需求的預估，設定下列ContainerGroupDefinition屬性：

- `TotalMemoryLimitMebibytes` – 設定容器群組的記憶體上限。群組中的所有容器都會共用配置的記憶體。如果您設定個別容器限制，總記憶體限制必須等於或大於最高容器特定記憶體限制。

- `TotalVcpuLimit` – 設定容器群組的 vCPU 上限。群組中的所有容器都會共用配置的 CPU 資源。如果您設定個別容器限制，總 CPU 限制必須等於或大於所有容器特定 CPU 限制的總和。最佳實務是考慮將此值設定為容器 CPU 限制總和的兩倍。

範例藍本

假設我們正在定義具有下列三個容器的遊戲伺服器容器群組：

- 容器 A 是我們的遊戲伺服器容器。我們估計一個遊戲伺服器在 512 MiB 和 1024 CPU 的資源需求。我們計劃讓容器執行 1 個伺服器程序。由於此容器會執行我們最重要的軟體，因此我們不設定記憶體限制或 vCPU 預留限制。
- 容器 B 是支援容器，其資源需求估計為 1024 MiB 和 1536 CPU。我們設定 2048 MiB 的記憶體限制，以及 1024 CPU 的 CPU 預留限制。
- 容器 C 是另一個支援容器。我們設定硬記憶體限制為 512 MiB，CPU 預留限制為 512 CPU。

使用此資訊，我們為容器群組設定下列總限制：

- 總記憶體限制：7680 MiB。此值超過最高記憶體限制 (1024 MiB)。
- CPU 總限制：13312 CPU。此值超過 CPU 限制 (1024+512 CPU) 的總和。

指定必要容器

對於每個執行個體容器群組，請將每個容器指定為必要或非必要。每個執行個體容器群組必須至少有一個基本支援容器。基本容器會執行容器群組的重要工作。基本容器預期一律會執行。如果失敗，整個容器群組會重新啟動。

將每個容器的 `ContainerDefinition` 屬性 `Essential` 設定為 `true` 或 `false`。

設定網路連線

您可以自訂網路存取，以允許外部流量連接到容器機群中的任何容器。例如，您必須建立與執行遊戲伺服器程序之容器的網路連線，讓遊戲用戶端可以加入並玩遊戲。遊戲用戶端會使用連接埠和 IP 地址連線至遊戲伺服器。

在容器機群中，用戶端和伺服器之間的連線不是直接的。在內部，容器中的程序會接聽容器連接埠。在外部，傳入流量會使用連線連接埠連線至機群執行個體。Amazon GameLift Servers 會維護內部容器連接埠與對外連線連接埠之間的映射，以便傳入流量路由至執行個體上的正確程序。

Amazon GameLift Servers 為您的網路連線提供額外的控制層。每個容器機群都有傳入許可設定，可讓您控制對每個對外連線連接埠的存取。例如，您可以移除所有連線連接埠的許可，以關閉機群容器的所有存取。

您可以更新機群的傳入許可、連線連接埠和容器連接埠。

Warning

如果您提供自訂 InstanceConnectionPortRange 或 InstanceInboundPermissions，Amazon GameLift Servers 將不再管理機群的任一值。您必須設定兩個欄位，以避免未定義的行為。

設定容器連接埠範圍

將容器連接埠範圍設定為每個容器定義的一部分。這是容器群組定義的必要參數。您需要設定足夠的連接埠，以容納所有同時執行且需要外部存取的程序。有些容器不需要任何連接埠。

執行遊戲伺服器的遊戲伺服器容器需要一個連接埠，才能用於每個同時執行的遊戲伺服器程序。遊戲伺服器程序會監聽指派的連接埠，並將其回報給 Amazon GameLift Servers。

設定連線連接埠範圍

使用一組連線連接埠設定容器機群。連線連接埠提供執行您容器之機群執行個體的外部存取權。Amazon GameLift Servers 會指派連線連接埠，並視需要將其映射至容器連接埠。

根據預設，會 Amazon GameLift Servers 計算所有容器群組所需的連接埠數目，並設定連接埠範圍以容納它們。我們強烈建議您使用 Amazon GameLift Servers 計算值，這些值會在您將更新部署至容器群組定義時更新。如果您需要自訂連線連接埠範圍，請使用下列指引。

當您建立容器機群時，請定義連線連接埠範圍（請參閱 [ContainerFleet : InstanceConnectionPortRange](#)）。請確定範圍有足夠的連接埠可對應至機群中兩個容器群組中所有容器所定義的每個容器連接埠。若要計算所需的最小連線連接埠，請使用下列公式：

$$[\text{Total number of container ports defined for containers in the game server container group}] * [\text{Number of game server container groups per instance}] + [\text{Total number of container ports defined for containers in the per-instance container group}]$$

最佳實務是連接埠數量下限的兩倍。

Note

連線連接埠的數量可能會限制每個執行個體的遊戲伺服器容器群組數量。如果機群每個執行個體只有一個遊戲伺服器容器群組的足夠連線連接埠，則只會 Amazon GameLift Servers 部署一個遊戲伺服器容器群組，即使執行個體具有多個遊戲伺服器容器群組的足夠運算能力。

設定傳入許可

傳入許可透過指定要為傳入流量開啟的連線連接埠，來控制容器機群的外部存取。您可以使用此設定，視需要開啟和關閉機群的網路存取。

根據預設，會Amazon GameLift Servers計算所有容器群組所需的連接埠數目，並設定連接埠範圍以容納它們。強烈建議您使用Amazon GameLift Servers計算值，這些值會在您將更新部署至容器群組定義時更新。如果您需要自訂連線連接埠範圍，請使用下列指引。

當您建立容器機群時，請定義一組傳入許可（請參閱 [ContainerFleet : InstanceInboundPermissions](#)）。傳入許可連接埠應與機群的連線連接埠範圍相符。

Note

由於容器連接埠是從 InstanceConnectionPortRange 隨機選取，為了確保可以進行工作階段連線，InstanceConnectionPortRange 中的所有連接埠都應涵蓋在 InstanceInboundPermissions 中的連接埠

範例藍本

此範例說明如何設定所有三個網路連線屬性。

- 我們機群的遊戲伺服器容器群組有 1 個容器，可執行 1 個遊戲伺服器程序。

在遊戲伺服器容器群組定義中，我們會設定此容器的 PortConfiguration 參數，如下所示：

```
"PortConfiguration": {
  "ContainerPortRanges": [ { "FromPort": 10, "ToPort": 20, "Protocol": "TCP" } ]
}
```

- 我們的機群也有一個具有 1 個容器的每個執行個體容器群組。它有 1 個程序需要網路存取。在每個執行個體容器定義中，我們設定此容器的 PortConfiguration 參數，如下所示：

```
"PortConfiguration": {
  "ContainerPortRanges": [ { "FromPort": 25, "ToPort": 25, "Protocol": "TCP" } ] }
```

- 我們的機群為每個機群執行個體設定了 20 個遊戲伺服器容器群組。根據此資訊，我們可以使用公式來計算所需的連線連接埠數量：
 - 下限：21 個連接埠 【1 個遊戲伺服器容器連接埠 * 每個執行個體 20 個遊戲伺服器容器群組 + 每個執行個體容器連接埠 1 個】

- 最佳實務：42 個連接埠 【最小連接埠 * 2】

建立容器機群時，我們會設定 InstanceConnectionPortRange 參數，如下所示：

```
"InstanceConnectionPortRange": { "FromPort": 1010, "ToPort": 1071 }
```

- 我們希望允許存取所有可用的連線連接埠。建立容器機群時，我們會設定 InstanceInboundPermissions 參數，如下所示：

```
"InstanceInboundPermissions": [  
  { "FromPort": 1010, "ToPort": 1071, "IpRange": "10.24.34.0/23", "Protocol":  
    "TCP" } ]
```

設定容器的運作狀態檢查

如果容器遇到終端故障並停止執行，容器會自動重新啟動。如果容器視為必要，它會提示整個容器群組重新啟動。

所有遊戲伺服器容器都會自動視為必要。支援容器可以指定為必要容器，但他們需要有機制來報告運作狀態。您也可以設定非必要支援容器的運作狀態檢查。

您可以定義其他自訂條件來測量容器運作狀態，並使用運作狀態檢查來測試該條件。若要設定容器運作狀態檢查，您可以在 Docker 容器映像或容器定義中定義它。如果您在容器定義中設定運作狀態檢查，它會覆寫容器映像中的任何設定。

為容器運作狀態檢查設定下列 SupportContainerDefinition 屬性：

- Command — 提供可檢查容器運作狀態某些層面的命令。您可以決定使用哪些條件來測量運作狀態。命令必須導致結束值為 1（運作狀態不佳）或 0（運作狀態良好）。
- StartPeriod — 指定運作狀態檢查失敗開始計數之前的初始延遲。此延遲可讓容器有時間引導其程序。
- Interval — 決定執行運作狀態檢查命令的頻率。您想要多快偵測和解決容器故障？
- Timeout — 決定等待成功或失敗的時間，然後再重試運作狀態檢查命令。運作狀態檢查命令需要多長時間才能完成？
- Retries — 在註冊失敗之前，應該重試多少次運作狀態檢查命令？

設定容器相依性

在每個容器群組中，您可以根據容器狀態在容器之間設定相依性。相依性會影響相依容器何時可以根據另一個容器的狀態啟動或關閉。

相依性的關鍵使用案例是為容器群組建立啟動和關閉序列。

例如，您可能希望容器 A 先啟動，並在容器 B 和 C 啟動之前成功完成。若要達成此目的，請先在容器 A 上建立容器 B 的相依性，條件為容器 A 必須成功完成。然後為容器 A 上具有相同條件的容器 C 建立相依性。啟動序列會以相反順序進行關機。

設定容器機群

當您建立容器機群時，請考慮下列決策點。這些點大多取決於您的容器架構和組態。

決定您要部署機群的位置

一般而言，您想要在玩家附近的地理位置部署機群，以將延遲降至最低。您可以將容器機群部署到任何 AWS 區域 Amazon GameLift Servers 支援的。如果您想要將相同的遊戲伺服器部署到其他地理位置，您可以將遠端位置新增至機群，包括 AWS 區域 和 Local Zones。對於多位置機群，您可以在每個機群位置獨立調整容量。如需支援的機群位置的詳細資訊，請參閱 [Amazon GameLift Servers 服務位置](#)。

考慮使用 [UDP ping 信標](#) 收集不同地理位置的網路延遲資料，以預測玩家裝置和潛在機群位置之間的延遲。這些特殊端點接受 UDP 訊息，而不是傳統的 ICMP ping，提供準確的延遲測量，以協助您選擇最佳的機群位置。

為您的機群選擇執行個體類型和大小

Amazon GameLift Servers 支援各種 Amazon EC2 執行個體類型，所有類型都可以與容器機群搭配使用。執行個體類型可用性和價格因位置而異。您可以在 Amazon GameLift Servers 主控台（在資源、執行個體和服務配額下）中檢視依位置篩選的支援執行個體類型清單。

選擇執行個體類型時，請先考慮執行個體系列。執行個體系列提供各種 CPU、記憶體、儲存和聯網功能的組合。取得 [EC2 執行個體系列](#) 的詳細資訊。在每個系列中，您有各種執行個體大小可供選擇。選取執行個體大小時，請考慮下列問題：

- 可支援工作負載的執行個體大小下限是多少？使用此資訊來消除任何過小的執行個體類型。
- 哪些執行個體類型大小非常適合您的容器架構？理想情況下，您想要選擇大小，以最小的浪費空間容納遊戲伺服器容器群組的多個複本。
- 擴展精細程度對遊戲有何意義？擴展機群容量涉及新增或移除執行個體，而每個執行個體代表託管特定數量遊戲工作階段的能力。考慮您要為每個執行個體新增或移除多少容量。如果玩家的需

求從分鐘到分鐘不等，那麼使用可以託管數百或數千個遊戲工作階段的非常大型執行個體可能很有意義。相反地，您可能偏好使用較小的執行個體類型進行更精細的擴展控制。

- 是否有根據大小節省成本？由於可用性，您可能會發現特定執行個體類型的成本因位置而異。

設定其他選用機群設定

您可以在設定容器機群時使用下列選用功能：

- 設定您的遊戲伺服器以存取其他 AWS 資源。請參閱 [與機群中的其他 AWS 資源通訊](#)。
- 保護具有作用中玩家的遊戲工作階段，避免在縮減規模事件期間提早終止。
- 限制一個人可以在有限的時間內在機群上建立的遊戲工作階段數量。

建立容器機群的Amazon GameLift Servers容器群組定義

容器群組定義說明如何將容器化遊戲伺服器應用程式部署到容器機群。這是一個藍圖，告知要部署到機群Amazon GameLift Servers的容器映像，以及如何執行它們。當您建立容器機群時，您可以指定要部署到機群的容器群組定義。如需容器群組的詳細資訊，請參閱 [容器機群元件](#)。

開始之前

開始建立容器群組定義之前該怎麼做的秘訣：

- 完成容器映像，並將其推送至您計劃建立容器群組的相同 AWS 區域 Amazon Elastic Container Registry (Amazon ECR) 儲存庫。會在您建立容器群組定義時Amazon GameLift Servers擷取每個映像的快照，並在部署至容器機群時使用快照。請參閱 [建置的容器映像 Amazon GameLift Servers](#)。
- 將您的容器定義建立為 JSON 檔案。容器群組定義包含一或多個容器定義。如果您使用 AWS CLI for 建立容器群組定義，則可以使用 JSON 檔案Amazon GameLift Servers。
- 確認您的 AWS 使用者具有存取 Amazon ECR 儲存庫的 IAM 許可。請參閱 [的 IAM 許可範例 Amazon GameLift Servers](#)。

建立遊戲伺服器容器群組定義

遊戲伺服器容器群組會執行您的遊戲伺服器軟體。遊戲伺服器容器群組有一個遊戲伺服器容器，可執行遊戲伺服器可執行檔。它也可以有一或多個支援容器來執行其他軟體，以支援您的遊戲伺服器。（這些有時稱為「附屬」容器。）

本主題說明如何使用Amazon GameLift Servers主控台或 CLI AWS 工具建立簡單的遊戲伺服器容器群組定義。如需選用功能的詳細資訊，請參閱 [自訂Amazon GameLift Servers容器機群](#)。

Note

您可以在建立大多數容器群組定義和容器定義設定之後進行變更。如果您變更容器定義，會 Amazon GameLift Servers 擷取已更新容器映像的新快照。

若要建立簡單的遊戲伺服器容器群組定義：

下列指示說明如何使用最少的必要參數和 Amazon GameLift Servers 預設值來建立容器群組定義。

Console

在 [Amazon GameLift Servers 主控台](#) 中，選取您要建立容器群組的 AWS 區域。

開啟主控台的左側導覽列，然後選擇受管容器：群組定義。在容器群組定義頁面上，選擇建立群組定義。

步驟 1：定義容器群組定義詳細資訊

1. 輸入容器群組定義名稱。名稱對於 AWS 帳戶 和 區域必須是唯一的。
2. 選取遊戲伺服器容器群組類型。
3. 針對總記憶體限制，輸入要提供給容器群組中所有容器的最大記憶體資源。如需計算此值的說明，請參閱 [設定資源限制](#)。
4. 針對總 vCPU 限制，輸入要提供給容器群組中所有容器的最大運算能力。如需計算此值的說明，請參閱 [設定資源限制](#)。

步驟 2：新增容器定義

遊戲伺服器容器群組至少有一個遊戲伺服器容器。在 主控台 中，您建立的第一個容器定義是遊戲伺服器容器。此步驟說明如何定義遊戲伺服器容器定義所需的最低設定。

1. 輸入容器定義名稱。為群組定義的每個容器都必須具有唯一的名稱值。
2. 使用遊戲伺服器建置連結到容器映像。輸入公有或私有儲存庫中容器映像的 Amazon ECR 映像 URI。您可以使用下列任何格式：
 - 僅限映像 URI：[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]
 - 映像 URI + 摘要：[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]@[digest]

- 映像 URI + 標籤：[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]:[tag]
3. 指定遊戲Amazon GameLift Servers伺服器建置使用的伺服器 SDK 版本。對於容器機群，此值必須是 5.2.0 或更高。
 4. 在內部容器連接埠範圍內，設定通訊協定並定義連接埠範圍。範圍大小必須大於在此容器中執行的並行遊戲伺服器程序數目。如果遊戲伺服器容器每個容器只執行一個伺服器程序，則此連接埠範圍只需要幾個連接埠。如需詳細資訊，請參閱[設定網路連線](#)。
 5. 視需要新增更多容器，以執行其他支援軟體。其他容器會自動指定支援容器。遊戲伺服器容器群組只能有一個遊戲伺服器容器和最多八個支援容器。提供下列最低必要設定：
 - 容器定義名稱
 - ECR 映像 URI。
 - 內部容器連接埠（只有在容器具有需要網路存取的程序時，才包含此選項。）

步驟 3：設定相依性

- 如果您的容器群組定義具有多個容器，您可以選擇在容器之間設定相依性。如需詳細資訊，請參閱[設定容器相依性](#)。

步驟 3：檢閱和建立

1. 檢閱所有容器群組定義設定。使用編輯來變更任何區段，包括群組的每個容器定義。
2. 檢閱完成後，請選擇建立。

如果您的請求成功，主控台會顯示新容器群組定義資源的詳細資訊頁面。一開始狀態為 **COPYING**，因為會Amazon GameLift Servers開始為群組拍攝所有容器映像的快照。當此階段完成時，容器群組定義狀態會變更為 **READY**。容器群組定義必須處於 **READY** 狀態，才能使用它建立容器機群。

AWS CLI

當您使用 AWS CLI 建立容器群組定義時，請將容器定義組態維護在個別的JSON檔案中。您可以在 CLI 命令中參考 檔案。如需結構描述範例[建立容器定義JSON檔案](#)，請參閱。

建立容器群組定義

若要建立新的容器群組定義，請使用 CLI `create-container-group-definition` 命令。如需此命令的詳細資訊，請參閱 AWS CLI 命令參考中的 [create-container-group-definition](#)。

此範例說明遊戲伺服器容器群組定義的請求。它假設您已建立具有此群組容器定義的 JSON 檔案。

```
aws gamelift create-container-group-definition \
  --name MyAdventureGameContainerGroup \
  --operating-system AMAZON_LINUX_2023 \
  --container-group-type GAME_SERVER \
  --total-memory-limit-mebibytes 4096 \
  --total-vcpu-limit 1 \
  --game-server-container-definition file://MyAdventureGameContainers.json
```

建立容器定義JSON檔案

建立容器群組定義時，您也可以定義群組的容器。容器定義會指定存放容器映像的 Amazon ECR 儲存庫，以及網路連接埠的選用組態、CPU 和記憶體用量限制，以及其他設定。我們建議為容器群組中的所有容器建立具有組態的單一 JSON 檔案。維護檔案對於存放、共用、追蹤這些重要組態的版本非常有用。如果您使用 AWS CLI 建立容器群組定義，您可以在 `命令中參考` 檔案。

建立容器定義

1. 建立並開啟新 .JSON 檔案。例如：

```
[~/work/glc]$ vim SimpleServer.json
```

2. 為群組的每個容器建立個別的容器定義。複製下列範例內容，並視需要修改容器。如需容器定義語法的詳細資訊，請參閱 Amazon GameLift Servers API 參考中的 [ContainerDefinitionInput](#)。
3. 將檔案儲存在本機，以便您可以在 CLI AWS 命令中參考。

範例：遊戲伺服器容器定義

Example

此範例說明遊戲伺服器容器群組的基本容器。基本複本容器包含您的遊戲伺服器應用程式、Amazon GameLift Servers 代理程式，並且可以包含其他支援遊戲託管的軟體。定義必須包含名稱、映像 URI 和連接埠組態。此範例也會設定一些容器特定的資源限制。


```
{
  "ContainerName": "MyAdventureGameServer",
  "ImageUri": "111122223333.dkr.ecr.us-east-1.amazonaws.com/gl-
containers:myadventuregame-server",
  "PortConfiguration": {
    "ContainerPortRanges": [
      {
        "FromPort": 2000,
        "Protocol": "TCP",
        "ToPort": 2010
      }
    ]
  },
  "ServerSdkVersion": "5.2.0"
}
```

更新容器機群的Amazon GameLift Servers容器群組定義

您可以更新容器群組定義的大部分屬性，包括個別容器定義。容器群組定義具有版本編號。當您更新容器群組定義時，會Amazon GameLift Servers儲存更新並遞增定義的版本編號。設定容器機群時，您可以指定要部署的容器群組定義版本。

更新容器群組定義後，您可以將新版本部署到新的或現有的容器機群。

更新遊戲伺服器容器群組定義

本主題說明如何使用Amazon GameLift Servers主控台或 CLI AWS 工具更新遊戲伺服器容器群組定義。如需選用功能的詳細資訊，請參閱 [自訂Amazon GameLift Servers容器機群](#)。

若要更新容器群組定義：

Console

在 [Amazon GameLift Servers主控台](#) 中，選取您要建立容器群組的 AWS 區域。

開啟主控台的左側導覽列，然後選擇受管容器：群組定義。在容器群組定義頁面上，選擇要更新的容器群組定義和版本。

儲存更新之後，您可以使用新版本來建立新的容器機群，也可以將更新部署到現有的容器機群。

步驟 1：定義容器群組定義詳細資訊

- 您可以更新總記憶體和 vCPU 限制設定。

步驟 2：新增容器定義

您可以進行下列容器定義更新：

- 更新現有的容器定義。
 - 新增支援容器定義。
 - 移除支援容器定義。
1. 您可以更新 ECR 映像 URI。請務必更新伺服器開發套件版本設定，以符合新映像。
 2. 您可以視需要更新內部容器連接埠範圍。當這些變更部署到機群時，您對這些設定所做的變更可能會影響容器機群的連線連接埠設定。如需詳細資訊，請參閱[設定網路連線](#)。

步驟 3：設定相依性

- 您可以視需要變更相依性。如需詳細資訊，請參閱[設定容器相依性](#)。

步驟 3：檢閱和建立

- 檢閱您的容器群組定義更新。使用編輯在任何區段中進行其他變更。完成後，請選擇建立以產生容器群組定義的新版本。

如果您的請求成功，主控台會顯示新容器群組定義資源的詳細資訊頁面。一開始狀態為 **COPYING**，因為會Amazon GameLift Servers開始為群組拍攝所有容器映像的快照。當此階段完成時，容器群組定義狀態會變更為 **READY**。容器群組定義必須處於 **READY** 狀態，然後才能使用它建立容器機群。

AWS CLI

當您使用 AWS CLI 建立或更新容器群組定義時，請將容器定義組態維護在個別的JSON檔案中。您可以在 CLI 命令中參考 檔案。如需結構描述範例[建立容器定義JSON檔案](#)，請參閱。

更新定義時，您只需指定要更新的值。會Amazon GameLift Servers保留您未包含在更新請求中的任何值。如果您要變更容器定義。不過，變更容器定義時，請提供完整的集合。

更新容器群組定義

若要更新新的容器群組定義，請使用 CLI `update-container-group-definition` 命令。如需此命令的詳細資訊，請參閱 AWS CLI 命令參考中的 [update-container-group-definition](#)。

Example：遊戲伺服器容器群組

您可以在擷取、更新或刪除容器群組定義時，或建立或更新容器機群時，指定容器群組定義版本。每個容器群組定義都有版本屬性。此外，和定義的 ARN 值會指定版本編號。

此範例說明遊戲伺服器容器群組定義變更的請求。其假設您已建立具有此群組容器定義的 JSON 檔案。此範例使用定義名稱的 ARN 值，並指定更新為版本 1。

```
aws gamelift update-container-group-definition \
  --name arn:aws:gamelift:us-west-2:111122223333:containergroupdefinition/
  MyAdventureGameContainerGroup:1 \
  --operating-system AMAZON_LINUX_2023 \
  --container-group-type GAME_SERVER \
  --total-memory-limit-mebibytes 4096 \
  --total-vcpu-limit 1 \
  --container-definitions file://SimpleServer.json
```

複製容器群組定義

您可以使用 Amazon GameLift Servers 主控台來複製現有的容器群組定義。

複製容器群組

1. 在 [Amazon GameLift Servers 主控台](#) 中，前往左側導覽窗格，然後選擇容器群組。
2. 在容器群組清單頁面上，選取要複製的現有容器群組。選取容器群組後，複製按鈕會處於作用中狀態。
3. 選擇複製。此動作會使用預先填入的設定開啟容器群組建立精靈。
4. 輸入複製容器群組的新名稱。相同區域中的容器群組必須具有唯一的名稱。
5. 逐步瀏覽容器群組和容器定義頁面，檢閱和建立新的容器群組。

刪除容器機群的 Amazon GameLift Servers 容器群組定義

您有多種刪除容器群組定義的選項。當您刪除容器群組定義時，此動作也會刪除容器群組中的所有容器定義。

容器群組定義可以有多個版本。容器群組版本具有相同的名稱，但具有不同的版本編號。容器群組定義 ARNs 指定名稱和版本。

您可以在擷取、更新或刪除容器群組定義時，或在建立或更新容器機群時指定容器群組定義版本。每個容器群組定義都有版本屬性。此外，定義的 ARN 值會指定版本編號。

有幾種方法可以刪除容器群組定義：

- 您可以刪除特定定義的所有版本。
- 您可以刪除特定定義的一個特定版本。
- 您可以保留一些最新版本，並刪除特定定義的較舊版本。例如，您可以刪除第 5 版之前的所有版本。

只有在容器機群中未使用容器群組定義版本時，才能刪除該版本。

刪除容器群組定義

Console

在 [Amazon GameLift Servers 主控台](#) 中，選取您要建立容器群組的 AWS 區域。

開啟主控台的左側導覽列，然後選擇受管容器：群組定義。在容器群組定義頁面上，選取要修改的定義，然後選擇刪除。

系統會提示您選取要進行的刪除類型，並根據刪除類型指定其他設定。

AWS CLI

- 若要刪除容器群組定義，請使用 `delete-container-group-definition` CLI 命令，並針對您要執行的刪除類型提供值。如需此命令的詳細資訊，請參閱 AWS CLI 命令參考中的 [delete-container-group-definition](#)。

此範例說明刪除版本 5 之前遊戲伺服器容器群組定義所有版本的請求。

Example

```
aws gamelift delete-container-group-definition \
  --name MyAdventureGameContainerGroup \
  --version-count-to-retain 5 \
```

Amazon GameLift Servers Anywhere 機群

當您想要利用自己的託管資源來利用 Amazon GameLift Servers 功能時，請使用機群 Anywhere。Anywhere 機群通常用作反覆開發的測試環境，或混合託管解決方案中的受管機群。

Anywhere 機群包含一組您提供和管理的運算資源（虛擬或實體）。運算可以位於任何具有連線能力的地理位置，從本機筆記型電腦到遠端前哨站。設定 Anywhere 機群時，您可以透過註冊運算，將運算新增至機群 Amazon GameLift Servers。每個運算都會使用其 IP 地址（或 DNS 名稱）註冊，讓 Amazon GameLift Servers 可以與其建立連線。

您可以在每個運算上安裝遊戲伺服器軟體並啟動遊戲伺服器程序，藉此將遊戲伺服器軟體部署到 Anywhere 機群。每個啟動的遊戲伺服器程序都會建立 Amazon GameLift Servers 服務的連線，並報告託管遊戲工作階段的準備程度。您可以使用現有的組態管理和部署工具來處理初始部署和主機管理任務。與搭配使用時，還需要一些額外的任務 Amazon GameLift Servers，包括：

- 註冊和取消註冊運算，以從機群新增或移除它們。
- 在所有運算上維持 up-to-date 身分驗證字符。運算上的伺服器程序會在連線至 Amazon GameLift Servers 服務時使用它們。

Note

選擇性地使用 Amazon GameLift Servers 代理程式部署您的 Anywhere 機群，以自動化這些金鑰管理任務。請參閱 [使用 Amazon GameLift Servers 代理程式](#)。

請參閱這些主題，了解如何設定和維護 Anywhere 機群：

- [使用 Amazon GameLift Servers Anywhere 託管的開發藍圖](#)
- [使用 混合託管的開發藍圖 Amazon GameLift Servers](#)
- [使用 Amazon GameLift Servers Anywhere 設定反覆開發](#)
- [建立 Amazon GameLift Servers Anywhere 機群](#)
- [Amazon GameLift Servers 機群建立的運作方式](#)
- [更新 Amazon GameLift Servers 機群組態](#)

Anywhere 機群建立工作流程

對於 Anywhere 機群，僅Amazon GameLift Servers設定機群資源。您設定並向機群註冊運算，並安裝遊戲伺服器軟體並啟動遊戲伺服器程序來託管遊戲工作階段。

1. Amazon GameLift Servers 在機群的主區域中建立機群資源。機群狀態和自訂位置狀態設定為新增。
2. Amazon GameLift Servers 開始將事件寫入機群事件日誌。
3. 建立機群資源後，Amazon GameLift Servers會將機群狀態設定為作用中。此時，您可以向機群註冊新的運算。

建立 Amazon GameLift ServersAnywhere 機群

本主題說明如何建立 Amazon GameLift ServersAnywhere 機群。透過 Anywhere 機群，您可以使用核心Amazon GameLift Servers遊戲工作階段管理功能，同時透過您自己的運算資源託管遊戲工作階段。為您的內部部署硬體或其他雲端型資源建立 Anywhere 機群。

Anywhere 機群通常與混合託管解決方案中的Amazon GameLift Servers受管機群一起使用。在開發使用託管的遊戲時，它們也提供有用的測試環境Amazon GameLift Servers。請參閱這些主題，進一步了解何時以及如何將 Amazon GameLift ServersAnywhere 機群納入遊戲託管解決方案：

- [混合託管](#)
- [使用 設定託管機群 Amazon GameLift Servers](#)
- [使用 Amazon GameLift ServersAnywhere 設定反覆開發](#)

由於 Anywhere 機群是自我管理的，因此設定機群需要額外的的工作。若要讓 Anywhere 機群準備好託管遊戲工作階段和玩家，您需要完成下列任務：

主題

- [開始之前](#)
- [建立自訂位置](#)
- [建立 Anywhere 機群](#)
- [將運算新增至機群](#)
- [啟動遊戲伺服器](#)

開始之前

在建立 Anywhere 機群之前，請執行下列任務。如需更詳細的指引，請參閱 [使用 Amazon GameLift Servers Anywhere 託管的開發藍圖](#) 或 [使用 混合託管的開發藍圖 Amazon GameLift Servers](#)。

- 將您的遊戲伺服器程式碼與Amazon GameLift Servers伺服器 SDK 5.x 版（或更新版本）整合。您不需要完成所有遊戲整合任務，只需要遊戲伺服器建置所需的任務。常見的做法是將本機機器設定為 Anywhere 機群，並使用命令列界面來測試您的遊戲伺服器整合（請參閱 [使用 Amazon GameLift Servers Anywhere 設定本機測試](#)）。您可以在開發時納入其他元件（例如Amazon GameLift Servers 已啟用的遊戲用戶端）。
- 封裝遊戲伺服器軟體以安裝到 Anywhere 機群運算。套件應包含您的整合遊戲伺服器建置，以及執行遊戲伺服器所需的所有支援軟體。
- 決定是否將 Amazon GameLift Servers 代理程式與 Anywhere 機群搭配使用。代理程式是一種運算上程序管理工具，可自動化與管理伺服器程序和運算相關的一些關鍵任務，以搭配使用Amazon GameLift Servers。如需詳細資訊，請參閱[使用 Amazon GameLift Servers 代理程式](#)。

建立自訂位置

建立自訂位置來代表運算資源的實體位置。建立 Anywhere 機群時，您必須至少已定義一個自訂位置。您可以隨時建立其他自訂位置，並將其新增至現有的機群。

建立自訂位置

使用 Amazon GameLift Servers主控台或 AWS Command Line Interface (AWS CLI) 來建立自訂位置。

Console

在 [Amazon GameLift Servers主控台](#) 中，使用導覽窗格開啟位置頁面。選擇建立位置以開啟建立對話方塊。

1. 在對話方塊中，輸入位置名稱。最佳實務是使用描述一組運算資源之有意義的位置的名稱。它可能是地理位置、資料中心名稱或其他位置識別符。會Amazon GameLift Servers附加自訂位置的名稱與 custom-。
2. （選用）將標籤新增至您的自訂位置。每個標籤皆包含由您定義的一個金鑰與一個選用值。將標籤指派給您想要以有用方式分類 AWS 的資源，例如依用途、擁有者或環境。針對您要新增的每個標籤，選擇新增標籤。
3. 選擇 Create (建立)。

AWS CLI

使用 [create-location](#) 命令建立自訂位置。提供 location-name 值，其開頭必須為 custom-。最佳實務是使用描述一組運算資源之有意義的位置的名稱。它可能是地理位置、資料中心名稱或其他位置識別符。

```
aws gamelift create-location \  
  --location-name custom-location-1
```

輸出

```
{  
  "Location": {  
    "LocationName": "custom-location-1",  
    "LocationArn": "arn:aws:gamelift:us-east-1:111122223333:location/custom-  
location-1"  
  }  
}
```

建立 Anywhere 機群

為您擁有的一組運算資源建立 Anywhere 機群。新的 Anywhere 機群開始清空；您可以透過註冊將運算新增至機群。

建立時，新的 Anywhere 機群會快速地在機群狀態中從 移至 NEW ACTIVE。您可以在機群達到 之後，將運算新增至機群ACTIVE。

建立 Anywhere 機群

使用 Amazon GameLift Servers主控台或 AWS Command Line Interface (AWS CLI) 來建立 Anywhere 機群。

Console

在 [Amazon GameLift Servers主控台](#) 中，使用導覽窗格開啟機群頁面。選擇建立機群以開始機群建立工作流程。

步驟 1 選擇運算類型

選取 Anywhere 選項，然後選擇下一步。

步驟 2 定義機群詳細資訊

在此步驟中，指定一些金鑰的全機群設定。

1. 填寫機群詳細資訊區段：
 - a. 輸入機群名稱。我們建議您使用機群命名模式，以便在檢視機群清單時更輕鬆地識別機群類型。
 - b. 提供機群的簡短描述。
2. 視需要設定這些選用的其他詳細資訊。您可以稍後更新這些機群設定。
 - a. 建立用於生產或生產前測試的機群時，請使用此設定來指定機群運算的每小時成本值。Amazon GameLift Servers 可以在遊戲工作階段置放程序期間使用此資訊，根據成本選取託管資源。
 - b. 如果您想要合併此機群和其他機群的指標資料，請指定指標群組名稱。針對您要合併的所有機群使用相同的指標群組名稱。檢視指標群組的指標，以查看彙總的資料。
3. 將選用標籤新增至您的自訂位置。每個標籤皆包含由您定義的一個金鑰與一個選用值。將標籤指派給您想要以有用方式分類 AWS 的資源，例如依用途、擁有者或環境。針對您要新增的每個標籤，選擇新增標籤。
4. 選擇下一步以繼續工作流程。

步驟 3 選取自訂位置

在此步驟中，識別您計劃新增至此機群的運算實體位置。您現在可以指定一或多個位置，也可以稍後視需要新增或移除位置。

1. 在自訂位置中，為機群的運算選取一或多個位置。此清單包含目前所選中定義的所有自訂位置 AWS 區域。若要定義您要新增至機群的新自訂位置，請選擇建立位置。
2. 選擇下一步以繼續工作流程。

步驟 4 檢閱和建立

建立機群之前，請先檢閱您的設定。

當您準備好部署新機群時，請選擇建立。會 Amazon GameLift Servers 立即開始機群啟用程序，指派唯一的 ID 並將機群置於 NEW 狀態。您可以在機群頁面上追蹤機群的進度。

AWS CLI

使用 [create-fleet](#) 命令來建立運算類型 的機群ANYWHERE。提供名稱和至少一個自訂位置。會在您目前的預設值中Amazon GameLift Servers建立 Anywhere 機群資源 AWS 區域（或者您可以新增 `--region` 標籤以指定不同的 AWS 區域）。

下列範例請求會使用最少的必要設定建立新的機群。*custom-location* 使用您自己的資訊取代 *FleetName* 和 。

```
aws gamelift create-fleet \  
--name FleetName \  
--compute-type ANYWHERE \  
--locations "Location=custom-location"
```

回應範例

```
{  
  "FleetAttributes": {  
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "FleetArn": "arn:aws:gamelift:us-west-2:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "Name": "HardwareAnywhere",  
    "CreationTime": "2023-02-23T17:57:42.293000+00:00",  
    "Status": "ACTIVE",  
    "MetricGroups": [  
      "default"  
    ],  
    "CertificateConfiguration": {  
      "CertificateType": "DISABLED"  
    },  
    "ComputeType": "ANYWHERE"  
  }  
}
```

建立時，新的 Anywhere 機群會快速移至機群狀態 ACTIVE。您可以在機群達到 之後，將運算新增至機群ACTIVE。

請注意，回應不包含機群位置。您可以呼叫 [describe-fleet-attributes](#) 和 來擷取完整的機群詳細資訊[describe-fleet-location-attributes](#)。

將運算新增至機群

若要將運算資源新增至機群，並準備好託管遊戲工作階段，請執行下列任務：

- 向機群註冊運算。註冊會告知Amazon GameLift Servers哪些實體託管資源是機群的一部分。
- 請求運算的身分驗證字符。在運算上執行的每個遊戲伺服器都需要此字符才能連線到Amazon GameLift Servers服務。身分驗證字符是暫時的，必須定期重新整理。

Note

如果您要使用 Amazon GameLift Servers代理程式部署遊戲伺服器軟體，可以略過此步驟。代理程式會自動註冊每個運算，並維護運算的有效身分驗證字符。請參閱 [使用 Amazon GameLift Servers 代理程式](#)。

您可以使用 CLI AWS 註冊運算並請求身分驗證字符，或對適用於的 AWS SDK 進程式設計呼叫 Amazon GameLift Servers。這些動作無法透過 Amazon GameLift Servers主控台使用。

最佳實務是，建議將啟動指令碼新增至每個運算，以自動化這兩個任務。啟動指令碼會自動呼叫 `register-compute` 和 `get-compute-auth-token` 命令。您也可以自動執行任務，在運算生命週期內定期重新整理身分驗證權杖，並在關閉時取消註冊運算。

每個啟動動作都會傳回您需要存放在運算上的運算特定值。當遊戲伺服器程序在運算上啟動時，在初始化與服務的連線時，必須將這些值作為伺服器參數傳遞 Amazon GameLift Servers (請參閱伺服器 SDK 參考 [ServerParameters](#) 中的)。建議您將這些運算特定值 (或其儲存位置) 設定為環境變數。如果您使用的是 Amazon GameLift Servers代理程式，則會為您處理此任務。運算特定的值如下：

- `register-compute` 會傳回的值 `GameLiftServiceSdkEndpoint`。將此值設定為 `webSocketUrl` 伺服器參數。
- `compute-auth-token` 會傳回身分驗證字符。將此值設定為 `authToken` 伺服器參數。

AWS CLI

下列指示說明如何使用 CLI AWS 手動提交每個請求。

註冊運算

呼叫 [register-compute](#) 註冊運算。識別要新增運算的機群 ID。提供下列運算資訊：有意義的名稱、IP 地址和位置。運算的位置必須是已與機群建立關聯的自訂位置。如果您想要使用不同的自訂

位置，請使用 Amazon GameLift Servers 主控台更新機群，或呼叫 AWS CLI 命令 [create-fleet-locations](#) 將自訂位置新增至機群。

在下列範例中，取代運算和機群的預留位置值。值 `fleet-id` 會在您建立 Anywhere 機群時傳回。您可以呼叫 [describe-fleet-attributes](#) 和 [describe-fleet-location-attributes](#) 來擷取完整的機群詳細資訊。

```
aws gamelift register-compute \  
  --compute-name HardwareAnywhere \  
  --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --ip-address 10.1.2.3 \  
  --location custom-location-1
```

範例輸出

```
{  
  "Compute": {  
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "FleetArn": "arn:aws:gamelift:us-west-2:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "ComputeName": "HardwareAnywhere",  
    "ComputeArn": "arn:aws:gamelift:us-west-2:111122223333:compute/  
HardwareAnywhere",  
    "IpAddress": "10.1.2.3",  
    "ComputeStatus": "Active",  
    "Location": "custom-location-1",  
    "CreationTime": "2023-02-23T18:09:26.727000+00:00",  
    "GameLiftServiceSdkEndpoint": "wss://us-west-2.api.amazongamelift.com"  
  }  
}
```

請求身分驗證字符

呼叫 [get-compute-auth-token](#) 以請求有效的身分驗證字符。註冊運算。識別機群 ID 和運算名稱。

在下列範例中，取代運算和機群的預留位置值。值 `fleet-id` 會在您建立 Anywhere 機群時傳回。您可以呼叫 [describe-fleet-attributes](#) 來擷取完整的機群詳細資訊。若要尋找運算資訊，[list-compute](#) 請使用機群 ID 呼叫，以查看已註冊至機群的所有運算。

```
aws gamelift get-compute-auth-token \  
  --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --compute-name HardwareAnywhere
```

```
--fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
--compute-name HardwareAnywhere
```

範例輸出

```
{  
  "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
  "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
  "ComputeName": "HardwareAnywhere",  
  "ComputeArn": "arn:aws:gamelift:us-east-1:111122223333:compute/  
HardwareAnywhere",  
  "AuthToken": "0c728041-3e84-4aaa-b927-a0fb202684c0",  
  "ExpirationTimestamp": "2023-02-23T18:47:54+00:00"  
}
```

啟動遊戲伺服器

在您建立 Anywhere 機群並將一或多個運算新增至機群之後，您就可以開始執行遊戲伺服器。

步驟 1 安裝遊戲伺服器軟體

在您的 Anywhere 機群中的每個運算上安裝遊戲伺服器組建和所有相依軟體。遊戲伺服器組建必須與 Amazon GameLift Servers 伺服器 SDK 5.x 版（或更新版本）整合，具有與服務通訊所需的最低功能。Amazon GameLift Servers

步驟 2 讓運算準備好執行遊戲伺服器

確保每個運算都已註冊，並具有有效的身份驗證字符。如果您使用指令碼來管理這些任務，請確定指令碼在每個運算上執行，然後再啟動任何遊戲伺服器程序。

如果您已使用遊戲伺服器軟體部署 Amazon GameLift Servers 代理程式，請確定代理程式可執行檔啟動。

步驟 3 啟動遊戲伺服器程序

在運算上執行遊戲伺服器可執行檔的執行個體。如果您的遊戲伺服器建置已正確整合，遊戲伺服器程序會使用 `InitSDK()` 一組有效的伺服器參數呼叫伺服器 SDK 動作。當伺服器程序準備好託管遊戲工作階段時，它會呼叫 `ProcessReady()`。

Note

如果您使用 Amazon GameLift Servers 代理程式部署遊戲伺服器軟體，則可以略過此步驟。代理程式會根據您提供的執行時間指示，自動啟動遊戲伺服器程序。

您可以透過檢視啟用和作用中伺服器程序的伺服器程序指標來監控進度。請參閱 [機群的 Amazon GameLift Servers 指標](#)。如果您的遊戲伺服器程序無法初始化，請確認該程序正在擷取其執行所在運算的正確伺服器參數值。

使用別名抽象機Amazon GameLift Servers群指定

Amazon GameLift Servers 別名用於抽象化託管目的地。託管目的地會告訴 Amazon GameLift Servers 在哪裡尋找可用資源來為玩家託管新的遊戲工作階段。在下列案例中，別名很有用：

- 如果您的遊戲不使用多機群佇列進行遊戲工作階段放置，它會指定Amazon GameLift Servers機群 ID 來請求新的遊戲工作階段。在遊戲生命週期內，您將多次取代機群、更新伺服器建置、更新託管硬體和作業系統，或解決效能問題。使用別名來抽象化機群 ID，讓您可以順暢地將玩家流量從現有機群切換到新的機群。
- 如果您想要在遊戲用戶端請求新遊戲工作階段時，執行其他動作，而非建立新的遊戲工作階段。例如，您可能想要將使用out-of-date用戶端的玩家導向升級網站。

別名必須指定路由策略。有兩種類型。簡單路由策略會將玩家流量路由到指定的機群 ID，您可以更新以重新導向流量。終端機路由策略會將訊息傳遞回用戶端，而不是建立新的遊戲工作階段。您可以隨時變更別名的路由策略。

如果您使用佇列放置遊戲工作階段，則不需要別名，即可在取代機群時重新導向流量。使用佇列，您可以直接新增機群並移除舊機群。玩家看不到此動作，因為新的遊戲工作階段請求會使用新的機群自動完成。它不會影響現有的遊戲工作階段。您可以使用機群 ID 或別名來識別佇列目的地。

主題

- [建立Amazon GameLift Servers別名](#)
- [編輯別名](#)

建立Amazon GameLift Servers別名

本主題說明如何建立Amazon GameLift Servers別名，以搭配遊戲工作階段放置使用。

建立別名

使用 Amazon GameLift Servers主控台或 AWS Command Line Interface (AWS CLI) 來建立別名。

Console

在 [Amazon GameLift Servers主控台](#) 中，使用導覽窗格開啟別名頁面。

1. 選擇建立別名。
2. 輸入別名名稱。我們建議您在別名名稱中包含有意義的特性，以便在檢視別名清單時提供協助。
3. 視需要輸入別名描述。
4. 選擇別名的路由策略。
 - a. 如果您選擇簡易路由策略，請從清單中選擇要與此別名建立關聯的機群 ID。此清單包含目前選取之 `eu` 中的所有機群 AWS 區域。您必須在與機群相同的區域中建立別名。
 - b. 如果您選擇終端機路由策略，請輸入Amazon GameLift Servers要傳回遊戲用戶端的字串值，以回應遊戲工作階段請求。具有終端機別名的請求會擲回內嵌訊息的例外狀況。
5. （選用）將標籤新增至別名資源。每個標籤皆包含由您定義的一個金鑰與一個選用值。將標籤指派給您想要以有用方式分類 AWS 的資源，例如依用途、擁有者或環境。為您要新增的每個標籤選擇新增標籤。
6. 當您準備好部署新機群時，請選擇建立。

AWS CLI

使用 `create-alias` 命令來建立別名。。Amazon GameLift Servers 會在您目前的預設值中建立別名資源 AWS 區域（或者您可以新增 `--region` 標籤來指定不同的標籤 AWS 區域）。

至少要包含別名名稱和路由策略。對於簡單的路由策略，指定與別名位於相同區域中的機群 ID。對於終端機路由策略，請提供訊息字串。

編輯別名

您可以使用 Amazon GameLift Servers主控台或 AWS CLI 命令 `update-alias` 編輯別名。

本主題說明如何編輯Amazon GameLift Servers別名，以搭配遊戲工作階段放置使用。您可以進行下列編輯：

編輯別名

使用 Amazon GameLift Servers主控台或 AWS Command Line Interface (AWS CLI) 來編輯別名。

Console

在 [Amazon GameLift Servers主控台](#) 中，使用導覽窗格開啟別名頁面。

1. 選取您要編輯的別名，然後選擇編輯。如果您沒有看到要編輯的別名，請檢查目前選取的別名 AWS 區域。
2. 在編輯別名頁面上，您可以進行下列編輯：
 - 變更別名名稱。
 - 變更別名描述。
 - 將路由策略從簡單變更為終端機，或從終端機變更為簡單。
 - 對於具有簡單路由策略的別名，請變更別名相關聯的機群 ID。
 - 對於具有終端機路由策略的別名，請變更訊息文字。
3. 選擇 Save changes (儲存變更)。使用簡單的路由策略更新別名的機群 ID 時，最多可能需要 2 分鐘才能完成轉換。在此期間，新的遊戲工作階段置放可能會在舊機群上進行。

AWS CLI

使用 [update-alias](#) 命令來變更別名資源。您可以更新目前預設值中的別名資源 AWS 區域，也可以新增 `--region` 標籤來指定不同的 AWS 區域。

您可以變更下列屬性：

- 別名名稱。
- 別名描述。
- 路由策略類型。請務必提供新路由策略的機群 ID 或訊息字串。
- 現有簡單路由策略的機群 ID。機群 ID 必須與別名位於相同的區域。
- 現有終端機路由策略的訊息字串。

使用Amazon GameLift Servers佇列管理遊戲工作階段置放

遊戲工作階段佇列是Amazon GameLift Servers使用 搜尋可用遊戲伺服器並選擇它們來託管新遊戲工作階段的主要機制。佇列提供更有效率的方式來處理大量遊戲工作階段請求，並在多個託管資源機群中尋找這些請求的位置。如果您的託管解決方案使用多個機群，而且您正在處理大量請求，您可能需要佇列。

當您的遊戲想要為玩家啟動新的遊戲工作階段時，它會傳送置放請求給 Amazon GameLift Servers 服務，該服務會將它排入佇列。佇列的組態決定處理請求的時間和方式。處理置放請求時，Amazon GameLift Servers會搜尋一組機群，讓遊戲伺服器託管遊戲工作階段。當 Amazon GameLift Servers找到可用的遊戲伺服器並提示它啟動遊戲工作階段時，配置會成功。

主題

- [佇列特性](#)
- [建立遊戲工作階段佇列](#)
- [自訂遊戲工作階段佇列](#)
- [設定遊戲工作階段放置的事件通知](#)
- [教學課程：使用 Spot 執行個體建立 Amazon GameLift Servers佇列](#)

佇列特性

Amazon GameLift Servers 遊戲工作階段佇列是 AWS 雲端資源。您可以在任何 AWS 區域 Amazon GameLift Servers 支援的 中建立佇列（請參閱 [Amazon GameLift Servers 服務位置](#)）。遊戲工作階段置放請求會傳送到該位置，並在該位置進行處理。

使用佇列將遊戲工作階段放置自動化，可為遊戲開發人員和玩家提供顯著的好處。其中包含：

- 佇列提供「最佳可能」放置。處理遊戲工作階段置放請求時，佇列會使用 Amazon GameLift Servers FleetIQ 演算法，根據一組定義的偏好設定來排定置放的優先順序，包括成本、位置和玩家延遲。
- 佇列支援 Spot 機群，以協助降低遊戲託管成本。您可以使用 AWS Spot 機群設定佇列，這通常會大幅降低託管成本，以及隨需機群。由於低成本是置放的關鍵條件之一，因此佇列永遠可以利用成本差異。
- 佇列可以在高需求期間更快地放置新遊戲。透過設定具有多個機群的佇列，您可以為遊戲工作階段放置提供更靈活的選項。但是，當需求增加時，其他機群也會視需要提供備份容量。對於任何置放請求，如果 Amazon GameLift Servers 無法將遊戲工作階段放在最偏好的位置，它會自動繼續評估其他位置。

- 佇列可讓遊戲伺服器可用性更具彈性。可能會發生中斷。使用多機群佇列時，慢速或中斷並不會影響玩家存取您的遊戲。透過使用在不同 AWS 區域 和可用區域中具有容量的機群來設定佇列，您可以協助確保玩家總是找到要加入的遊戲工作階段。
- 取得遊戲工作階段置放和佇列效能的指標。Amazon GameLift Servers 會發出佇列指標，包括置放成功和失敗的統計資料、佇列中的請求數量，以及請求在佇列中花費的平均時間。您可以在 Amazon GameLift Servers 主控台或 CloudWatch 中檢視這些指標。

若要開始建立基本入門佇列，請參閱 [建立遊戲工作階段佇列](#)。

建立遊戲工作階段佇列

佇列用於跨多個機群和位置放置新的遊戲工作階段。您的遊戲會透過向佇列提交放置請求來啟動新的遊戲工作階段。佇列已設定有關如何處理請求的指示。進一步了解如何在 中啟動遊戲工作階段置放請求 [建立遊戲工作階段](#)。

建立遊戲工作階段佇列

這些指示說明如何以最少的組態設定和預設設定建立簡單的工作佇列。有多種選項可自訂佇列組態。這些選項可協助您根據遊戲的需求進行最佳的配置。若要進一步了解如何自訂遊戲的佇列，請參閱 [自訂遊戲工作階段佇列](#)。您可以隨時更新大多數佇列組態設定。

您可以使用 Amazon GameLift Servers 主控台或 AWS CLI 建立遊戲工作階段佇列。

Console

在 [Amazon GameLift Servers 主控台](#) 中，選取要使用的 AWS 區域。開啟主控台的左側導覽列，然後選擇佇列。

1. 在佇列頁面上，選擇建立佇列以啟動工作流程。
2. 在佇列設定下，輸入下列設定：
 - a. 輸入佇列名稱。此名稱對於您要建立佇列 AWS 區域 的 必須是唯一的。
 - b. 保留預設逾時設定，即 600 秒（或 10 分鐘）。此值控制 Amazon GameLift Servers 在停止之前嘗試放置新遊戲工作階段的時間長度。Amazon GameLift Servers 會搜尋可用的資源，直到請求逾時為止。您可以隨時更新佇列的逾時設定。
 - c. 略過玩家延遲政策區段。佇列只有在收到包含玩家延遲資料的置放請求時，才會使用延遲政策。您可以隨時將延遲政策新增至佇列。如需建立延遲政策的詳細資訊，請參閱 [建立玩家延遲政策](#)。

3. 略過遊戲工作階段置放位置區段，以使用所有位置的預設設定。此設定可讓您建立允許清單，列出佇列可以進行置放的位置（也稱為篩選條件組態）。如需依位置和篩選條件組態排定優先順序的詳細資訊，請參閱[依位置排定置放的優先順序](#)。
4. 在目的地順序下，將一或多個機群新增至佇列。您可以使用機群 IDs 或 ARNs，或使用機群別名來識別機群。新增多個機群時，請記住它們都應該執行類似的遊戲組建，並與使用此佇列的任何遊戲用戶端相容。此外，佇列中的所有機群都必須具有相同的憑證組態。
 - a. 選取建立機群或別名的區域。對於多位置機群，這是「首頁」區域。
 - b. 針對目的地類型，選取機群或別名。
 - c. 您的區域和類型選擇會填入現有機群或別名的下拉式清單。選取一個以指定為佇列目的地。
 - d. 若要為佇列指定其他機群或別名，請選擇新增目的地並重複上述步驟。
 - e. 新增目的地清單之後，請使用drag-and-drop功能來重新排序目的地。會在依目的地排定置放的優先順序時Amazon GameLift Servers使用此順序。
5. 略過遊戲工作階段置放優先順序區段，以保留預設優先順序。此設定可讓您自訂如何Amazon GameLift Servers選擇尋找新遊戲工作階段置放可用託管資源的位置。如需排定置放優先順序的詳細資訊，請參閱[排定遊戲工作階段置放的優先順序](#)。您可以隨時更新佇列的置放優先順序。
6. 在位置順序下，保留預設值。依機群位置排定優先順序時，會使用此設定。它提供要使用的位置順序。使用預設優先順序設定時，當偏好的目的地是具有多個位置的機群時，位置會用作繫結器。
7. 略過選用的事件通知設定區段。處理大量置放請求的佇列需要事件通知。對於處理低磁碟區的佇列，例如用於開發或測試目的，您可以透過輪詢[DescribeGameSessionPlacement](#)來追蹤置放請求的狀態。如需詳細資訊，請參閱[設定遊戲工作階段放置的事件通知](#)。您可以隨時更新佇列的事件通知設定。
8. 選擇建立以產生最少自訂的新佇列。

AWS CLI

Example 建立佇列

下列範例會建立具有這些組態的遊戲工作階段佇列：

- 五分鐘逾時。
- 兩個機群目的地。
- 篩選以僅允許在下列位置放置：us-east-1、us-east-2、us-west-2和 ca-central-1。

- 根據成本和指定順序中位置的優先順序。

```
aws gamelift create-game-session-queue \  
  --name "sample-test-queue" \  
  --timeout-in-seconds 300 \  
  --destinations DestinationArn="arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-772266ba-8c82-4a6e-b620-a74a62a93ff8" DestinationArn="arn:aws:gamelift:us-  
east-1:111122223333:fleet/fleet-33f28fb6-aa8b-4867-85b4-ceb217bf5994" \  
  --filter-configuration "AllowedLocations=us-east-1, ca-central-1, us-east-2, us-  
west-2" \  
  --priority-configuration PriorityOrder="COST","LOCATION",LocationOrder="us-  
east-1","us-east-2","ca-central-1","us-west-2" \  
  --notification-target "arn:aws:sns:us-east-1:111122223333:gamelift-test.fifo"
```

Note

您可以使用機群或別名 ID 來呼叫 [describe-fleet-attributes](#) 或 [describe-alias](#) 以取得機群與別名 ARN 值。

如果 `create-game-session-queue` 請求成功，Amazon GameLift Servers 即會傳回 [GameSessionQueue](#) 物件，其中包含新的佇列組態。您現在可以使用 [StartGameSessionPlacement](#) 來提交請求到佇列。

Example 使用玩家延遲政策建立佇列

下列範例會建立具有這些組態的遊戲工作階段佇列：

- 十分鐘逾時
- 三個機群目的地
- 一組玩家延遲政策

```
aws gamelift create-game-session-queue \  
  --name "matchmaker-queue" \  
  --timeout-in-seconds 600 \  
  --destinations DestinationArn=arn:aws:gamelift:us-east-1::alias/alias-a1234567-  
b8c9-0d1e-2fa3-b45c6d7e8910 \  
                  DestinationArn=arn:aws:gamelift:us-west-2::alias/alias-b0234567-  
c8d9-0e1f-2ab3-c45d6e7f8901 \  
  --notification-target "arn:aws:sns:us-east-1:111122223333:gamelift-test.fifo"
```

```
DestinationArn=arn:aws:gamelift:us-west-2::fleet/fleet-f1234567-
b8c9-0d1e-2fa3-b45c6d7e8912 \
--player-latency-policies
"MaximumIndividualPlayerLatencyMilliseconds=50,PolicyDurationSeconds=120" \
"MaximumIndividualPlayerLatencyMilliseconds=100,PolicyDurationSeconds=120" \
"MaximumIndividualPlayerLatencyMilliseconds=150" \
```

如果 `create-game-session-queue` 請求成功，Amazon GameLift Servers 即會傳回 [GameSessionQueue](#) 物件，其中包含新的佇列組態。

自訂遊戲工作階段佇列

本主題說明如何自訂遊戲工作階段佇列，以針對遊戲工作階段放置做出最佳可能決策。如需遊戲工作階段佇列及其運作方式的詳細資訊，請參閱 [使用Amazon GameLift Servers佇列管理遊戲工作階段置放](#)。

這些Amazon GameLift Servers功能需要佇列：

- [以 FlexMatch 進行配對](#)
- [設計 Spot 執行個體的佇列](#)

主題

- [Amazon GameLift Servers 遊戲工作階段佇列的最佳實務](#)
- [定義佇列的範圍](#)
- [建置多位置佇列](#)
- [排定遊戲工作階段置放的優先順序](#)
- [評估佇列指標](#)
- [設計 Spot 執行個體的佇列](#)

Amazon GameLift Servers 遊戲工作階段佇列的最佳實務

遊戲工作階段佇列包含Amazon GameLift Servers可以放置新遊戲工作階段的機群清單。每個機群都可以在多個地理位置部署託管資源。選擇置放時，佇列會根據您為機群設定的一組優先順序，選取機群和機群位置。

請考慮下列準則和最佳實務：

- 在涵蓋玩家的位置新增機群。您可以在任何可用位置新增機群和別名。如果您根據報告的玩家延遲進行放置，位置很重要。
- 對所有機群使用別名。將別名指派給佇列中的每個機群，並在佇列中設定目的地時使用別名名稱。
- 對所有機群使用相同或類似的遊戲組建或指令碼。佇列可能會將玩家放入佇列中任何機群的遊戲工作階段。玩家必須能夠在任何機群的任何遊戲工作階段中玩遊戲。
- 在至少兩個位置建立機群。透過在至少一個其他位置託管遊戲伺服器，您可以減輕區域中斷對玩家的影響。您可以保持備份機群縮減規模，並在用量增加時使用自動擴展來增加容量。
- 排定遊戲工作階段置放的優先順序。佇列會根據數個元素排定置放選擇的優先順序，包括目的地清單順序。
- 在與用戶端服務相同的位置建立佇列。透過將佇列放在用戶端服務附近的位置，您可以將通訊延遲降至最低。
- 使用具有多個位置的機群。使用佇列篩選條件組態，以防止佇列將遊戲工作階段放置在指定位置。您可以使用至少兩個具有不同主位置的多位置機群，以減輕區域中斷期間遊戲置放的影響。
- 對所有機群使用相同的 TLS 憑證設定。連線至機群中遊戲工作階段的遊戲用戶端必須具有相容的通訊協定。

定義佇列的範圍

您遊戲的玩家群體可能有一群不應一起玩的玩家。例如，如果您以兩種語言發佈遊戲，每種語言都應有自己的遊戲伺服器。

若要為您的玩家群體設定遊戲工作階段放置，請為每個玩家客群建立個別的佇列。調整每個佇列的範圍，將玩家放入正確的遊戲伺服器。限制佇列範圍的一些常見方法包括：

- 依地理位置。在多個地理區域部署遊戲伺服器時，您可以在每個位置為玩家建立佇列，以減少玩家延遲。
- 透過建置或指令碼變化。如果您的遊戲伺服器有多個變化，您可能支援無法在相同遊戲工作階段中玩的玩家群組。例如，遊戲伺服器建置或指令碼可能支援不同的語言或裝置類型。
- 依事件類型。您可以建立特殊佇列，為競賽或其他特殊活動的參與者管理遊戲。

設計多個佇列

根據您的遊戲和玩家，您可能想要建立多個遊戲工作階段佇列。當您的遊戲用戶端服務請求新的遊戲工作階段時，它會指定要使用的遊戲工作階段佇列。為了協助您決定是否使用多個佇列，請考慮：

- 遊戲伺服器的變體。您可以為遊戲伺服器的每個變化建立個別的佇列。佇列中的所有機群都必須部署相容的遊戲伺服器。這是因為使用佇列加入遊戲的玩家必須能夠在任何佇列的遊戲伺服器上玩遊戲。
- 不同的玩家群組。您可以自訂 如何根據玩家群組Amazon GameLift Servers放置遊戲工作階段。例如，您可能需要為需要特殊執行個體類型或執行時間組態的特定遊戲模式自訂佇列。或者，您可能想要特殊佇列來管理賽事或其他活動的配置。
- 遊戲工作階段佇列指標。您可以根據收集遊戲工作階段置放指標的方式來設定佇列。如需詳細資訊，請參閱[佇列的 Amazon GameLift Servers 指標](#)。

建置多位置佇列

我們建議對所有佇列進行多位置設計。此設計可以改善置放速度和託管彈性。必須使用多位置設計來使用玩家延遲資料，將玩家放入遊戲工作階段，並將延遲降至最低。如果您要建置使用 Spot 執行個體機群的多位置佇列，請遵循 中的指示[教學課程：使用 Spot 執行個體建立 Amazon GameLift Servers 佇列](#)。

建立多位置佇列的一種方法是將[多位置機群](#)新增至佇列。如此一來，佇列可以將遊戲工作階段放置在任何機群的位置。您也可以新增具有不同組態或主要位置的其他機群以進行備援。如果您使用的是多位置 Spot 執行個體機群，請遵循最佳實務，並包含具有相同位置的隨需執行個體機群。

下列範例概述設計基本多位置佇列的程序。在此範例中，我們使用兩個機群：一個 Spot 執行個體機群和一個隨需執行個體機群。每個機群 AWS 區域 都有下列放置位置：us-east-1、ca-central-1、us-east-2和 us-west-2。

使用多位置機群建立基本多位置佇列

1. 選擇要建立佇列的位置。您可以將佇列放置在部署用戶端服務附近的位置，將請求延遲降至最低。在此範例中，我們會在 中建立佇列us-east-1。
2. 建立新的佇列，並將多位置機群新增為佇列目的地。目的地順序會決定如何Amazon GameLift Servers放置遊戲工作階段。在此範例中，我們會先列出 Spot 執行個體機群，再列出隨需執行個體機群。
3. 定義佇列的遊戲工作階段置放優先順序。此順序會決定佇列先搜尋可用遊戲伺服器的位置。在此範例中，我們使用預設優先順序。
4. 定義位置順序。如果您未定義位置順序，會依字母順序Amazon GameLift Servers使用位置。

排定遊戲工作階段置放的優先順序

Amazon GameLift Servers 使用 演算法來判斷佇列目的地的優先順序，並判斷放置新遊戲工作階段的位置。演算法是以一組有序的條件為基礎。您可以使用預設優先順序，也可以自訂順序。您可以隨時編輯佇列的優先順序。

預設優先順序

1. 延遲 – 如果遊戲工作階段置放請求包含玩家的位置特定延遲資料，會Amazon GameLift Servers計算每個位置的平均玩家延遲，並嘗試將遊戲工作階段放置在平均最低的機群位置。
2. 成本 – 如果請求不包含延遲資料，或多個機群具有相同的延遲，則 會Amazon GameLift Servers評估每個機群的託管成本。機群的託管成本會根據機群類型 (Spot 或隨需)、執行個體類型和位置而有所不同。
3. 目的地 – 如果多個機群具有相同的延遲和成本，則 會根據佇列組態中列出的目的地順序來排定機群的Amazon GameLift Servers優先順序。
4. 位置 – 對於具有多位置機群的佇列，如果所有其他條件相等，則 會根據字母順序排定機群位置的Amazon GameLift Servers優先順序。

自訂佇列如何排定遊戲工作階段置放的優先順序

您可以選擇自訂佇列如何排定置放條件的優先順序。佇列會將自訂優先順序套用至其收到的所有遊戲工作階段置放請求。

Note

如果您建立自訂優先順序組態，但不包含所有四個條件，Amazon GameLift Servers會自動以預設順序附加任何缺少的條件。

自訂佇列的優先順序組態

使用 [Amazon GameLift Servers主控台](#) 或 AWS Command Line Interface (AWS CLI) 建立自訂優先順序組態。

Console

在 [Amazon GameLift Servers主控台](#) 中，您可以在建立新佇列或更新現有佇列時自訂佇列的優先順序。選取要使用的 AWS 區域。

開啟主控台的左側導覽列，然後選擇佇列。在佇列頁面上，選取現有的佇列，然後選擇編輯。

1. 前往遊戲工作階段置放優先順序一節。拖放每個優先順序條件以建立您想要的順序。
2. 前往位置順序一節。新增任何您想要排定優先順序的位置。當佇列具有具有多個位置的機群時，此清單很有用。您至少必須指定一個位置。您在此處指定的位置會優先排序，後面接著佇列目的地中的所有其他位置。
3. 選擇儲存變更。

AWS CLI

使用 [update-game-session-queue](#) 命令搭配 `--priority-configuration` 選項來自訂佇列的優先順序。會 Amazon GameLift Servers 更新目前預設區域中的佇列 AWS，或者您可以新增 `--region` 標籤來指定不同的 AWS 區域。

下列範例請求會新增或更新指定佇列的優先順序組態

```
aws gamelift update-game-session-queue \
  --name "example-queue-with-priority"
  --priority-configuration
  PriorityOrder="COST",'LOCATION',"DESTINATION",LocationOrder="us-east-1","us-
east-2","ca-central-1","us-west-2" \
```

依玩家延遲排定置放的優先順序

如果您想要為玩家提供最佳的玩家體驗，並確保最低延遲，請在設定遊戲工作階段放置系統時採取下列步驟：

- 將佇列設定為在選擇遊戲工作階段放置位置時優先考慮延遲。根據預設，延遲位於優先順序清單的最上層。您也可以自訂佇列的優先順序組態，並選擇將延遲置於優先順序的位置。
- 為您的佇列設定玩家延遲政策。延遲政策可讓您設定在遊戲工作階段置放中允許的延遲量硬性限制。如果 Amazon GameLift Servers 無法在未超過限制的情況下放置遊戲工作階段，放置請求將會逾時並失敗。您可以設定單一延遲政策，也可以建立一系列政策，隨著時間逐漸放寬延遲限制。透過一系列政策，您可以指定極低的初始延遲限制，並且在短暫延遲後仍可容納延遲較高的玩家。如需建立延遲政策的詳細資訊，請參閱 [建立玩家延遲政策](#)。
- 提出遊戲工作階段置放請求時（請參閱 [StartGameSessionPlacement](#)），請包含每個玩家的延遲資料。玩家延遲資料包含可能放置遊戲工作階段的每個可能位置的值。例如，對於將遊戲工作階段放置在 AWS 區域 us-east-2 和 ca-central-1 中的佇列，延遲資料可能如下所示：

```
"PlayerLatencies": [
  { "LatencyInMilliseconds": 100, "PlayerId": "player1", "RegionIdentifier": "us-east-2" },
  { "LatencyInMilliseconds": 100, "PlayerId": "player1", "RegionIdentifier": "ca-central-1" },
  { "LatencyInMilliseconds": 150, "PlayerId": "player2", "RegionIdentifier": "us-east-2" },
  { "LatencyInMilliseconds": 150, "PlayerId": "player2", "RegionIdentifier": "ca-central-1" }
]
```

若要取得準確的延遲測量，請使用 Amazon GameLift Servers 的 UDP ping 信標。這些端點可讓您測量玩家裝置與每個潛在託管位置之間的實際 UDP 網路延遲，從而做出比使用 ICMP ping 更準確的置放決策。如需使用 UDP ping 信標測量延遲的詳細資訊，請參閱 [UDP ping 信標](#)。

依位置排定置放的優先順序

您可以設定佇列，根據地理位置的優先順序清單進行遊戲工作階段置放。Location 是決定佇列如何選擇放置新遊戲工作階段位置的條件之一。根據預設，位置在延遲、成本和目的地之後會排在第四位。

對於遊戲工作階段放置，目的地和位置的含義略有不同：

- 目的地是指特定機群，並包含所有機群的託管資源，無論部署到何處。依目的地排定優先順序時，Amazon GameLift Servers 可能會在機群中的任何位置進行置放。多位置受管機群和 Anywhere 機群可以有部署到一或多個位置的託管資源。
- 位置是指部署機群託管資源的特定地理位置。機群可以有多個位置，其中可能包括 AWS 區域 Local Zones 或自訂位置（適用於 Anywhere 機群）。單一位置受管機群有一個位置，且一律為 AWS 區域。多位置受管機群具有主要區域，並且可以有遠端位置。Anywhere 機群有一或多個自訂位置。

依位置排定置放的優先順序時，Amazon GameLift Servers 會尋找任何包含優先順序位置的佇列目的地，並搜尋可用的託管資源。如果有多個具有優先順序位置的目的地，Amazon GameLift Servers 會移至下一個優先順序條件（成本、延遲、目的地）。

有多種方式可以影響佇列位置的優先順序

- 設定佇列如何處理所有遊戲工作階段置放請求：
 - 將優先順序組態新增至佇列。佇列的優先順序組態包含位置的排序清單。您可以指定一或多個要排定優先順序的位置。此清單不會排除任何位置，只會告訴先尋找可用託管資源 Amazon GameLift

Servers的位置。排序位置清單的常見用途是當您想要將大多數流量輸送至一或多個特定地理位置，並使用其他位置做為備份容量時。透過呼叫 [UpdateGameSessionQueue](#) 新增優先順序組態。

- 將篩選條件組態新增至佇列。篩選條件組態是佇列的允許清單。它會Amazon GameLift Servers告知在尋找可用的託管資源時忽略不在清單中的任何位置。篩選條件組態有兩種常見用途。首先，對於具有多個位置的機群，您可以使用篩選條件來排除部分機群的位置。其次，您可能想要暫時不允許在特定位置放置；例如，某個位置可能遇到暫時性問題。由於您可以隨時更新佇列的篩選組態，因此您可以視需要輕鬆新增和移除位置。透過呼叫 [UpdateGameSessionQueue](#) 新增篩選條件組態。
- 針對個別置放請求使用特殊指示：
 - 在遊戲工作階段置放請求中包含優先順序覆寫清單。您可以使用任何 [StartGameSessionPlacement](#) 請求提供位置的替代優先順序清單。此清單會針對僅針對該請求的位置，有效地取代佇列設定的優先順序。它不會影響任何其他請求。此覆寫功能有幾個需求：
 - 僅將覆寫清單與具有優先順序組態的佇列搭配使用，並以 LOCATION作為第一優先順序。
 - 請勿在相同的置放請求中包含玩家延遲資料。包含延遲資料會在排定Amazon GameLift Servers無法解析的位置優先順序時設定衝突。
 - 如果無法在優先順序覆寫清單中找到可用的資源，請決定Amazon GameLift Servers您要如何繼續。選擇返回佇列的其他位置，或限制覆寫清單的位置。根據預設，Amazon GameLift Servers會回到佇列的其他位置嘗試放置。
 - 視需要更新佇列的篩選條件組態，例如在覆寫清單上新增位置。覆寫清單不會使篩選條件清單失效。

建立玩家延遲政策

如果您的置放請求包含玩家延遲資料，會在所有玩家的平均延遲最低的位置Amazon GameLift Servers尋找遊戲工作階段。根據平均玩家延遲放置遊戲工作階段Amazon GameLift Servers可防止大多數玩家在高延遲的遊戲中放置。不過，Amazon GameLift Servers仍會放置具有極端延遲的玩家。若要容納這些玩家，請建立玩家延遲政策。

玩家延遲政策Amazon GameLift Servers可防止將請求的遊戲工作階段放置在玩家在請求中遇到延遲超過最大值的位置。玩家延遲政策也可以Amazon GameLift Servers防止將遊戲工作階段請求與較高延遲的玩家配對。

 Tip

若要管理延遲特定規則，例如在群組中的所有玩家之間需要類似的延遲，您可以使用 [Amazon GameLift ServersFlexMatch](#) 來建立以延遲為基礎的配對規則。

例如，請考慮此佇列具有 5 分鐘逾時和下列玩家延遲政策：

1. 花費 120 秒搜尋所有玩家延遲都小於 50 毫秒的位置。
2. 花費 120 秒搜尋所有玩家延遲都小於 100 毫秒的位置。
3. 花費剩餘的佇列時間，直到逾時為止，搜尋所有玩家延遲都小於 200 毫秒的位置。

評估佇列指標

使用指標來評估您的佇列執行情況是否良好。您可以在[Amazon GameLift Servers主控台](#)或 Amazon CloudWatch 中檢視與佇列相關的指標。如需佇列指標的清單和說明，請參閱 [佇列的 Amazon GameLift Servers 指標](#)。

佇列指標可以提供有關下列項目的洞見：

- 整體佇列效能 – 佇列指標指出佇列回應置放請求的成功程度。這些指標也可以協助您識別置放失敗的時間和原因。對於具有手動擴展機群的佇列，AverageWaitTime和 QueueDepth指標可以指出何時應調整佇列的容量。
- FleetIQ 演算法效能 – 對於使用FleetIQ演算法的置放請求，指標會顯示演算法找到理想遊戲工作階段置放的頻率。置放可能會優先使用具有最低玩家延遲的資源，或具有最低成本的資源。也有錯誤指標，可識別Amazon GameLift Servers找不到理想配置的常見原因。如需指標的詳細資訊，請參閱 [使用 Amazon CloudWatch 監控 Amazon GameLift Servers](#)。
- 位置特定置放 – 對於多位置佇列，指標會依位置顯示成功置放。對於使用FleetIQ演算法的佇列，此資料可提供玩家活動發生位置的實用洞見。

評估FleetIQ演算法效能的指標時，請考慮下列秘訣：

- 若要追蹤佇列尋找理想置放的速率，請使用 PlacementsSucceeded 指標搭配最低延遲和最低價格的FleetIQ指標。
- 若要提高佇列尋找理想置放的速率，請檢閱下列錯誤指標：

- 如果 FirstChoiceOutOfCapacity 很高，請調整佇列機群的容量擴展。
- 如果 FirstChoiceNotViable 錯誤指標很高，請查看您的 Spot 執行個體機群。當特定執行個體類型的中斷率太高時，Spot 執行個體機群會被視為不可行。若要解決此問題，請將佇列變更為使用具有不同執行個體類型的 Spot 執行個體機群。建議您在每個位置包含具有不同執行個體類型的 Spot 執行個體機群。

設計 Spot 執行個體的佇列

您可以使用 Spot 機群，大幅節省託管成本。如需詳細資訊，請參閱[隨需執行個體與 Spot 執行個體](#)。若要將 Spot 機群新增至您的託管解決方案，您需要使用 Spot 機群和隨需機群的組合來設定遊戲工作階段佇列。會在遊戲工作階段置放程序期間 Amazon GameLift Servers 使用佇列來搜尋多個機群，並尋找新遊戲工作階段的最佳可用主機。本主題提供如何開始使用 Spot 機群的指引。

您是否使用 FlexMatch 進行配對？您可以使用下列步驟，將 Spot 機群新增至現有的遊戲工作階段佇列，以進行配對配置。

1. 判斷遊戲工作階段佇列的目的地。

使用佇列管理遊戲工作階段置放是最佳實務，使用 Spot 執行個體時需要。由於 Spot 執行個體不一定在需要時可用，因此您需要設計彈性佇列，其中包含 Spot 機群和隨需機群，以提供備份容量。您可以保持隨需機群縮減規模，直到需要為止。若要設計佇列，請考慮下列事項：

- 位置 – 如果可能，您的 Spot 機群和隨需機群應與玩家位於相同的區域。將 Spot 資源和隨需資源放置在您要支援的每個位置。多位置機群同時支援 Spot 和隨需執行個體。
- 執行個體類型 – 考慮遊戲伺服器的硬體需求，以及您選擇的位置中執行個體的可用性。

若要嘗試最佳化 Spot 可用性和彈性的佇列，請參閱[教學課程：使用 Spot 執行個體建立 Amazon GameLift Servers 佇列](#)。如需 Spot 設計最佳實務，請參閱[Amazon GameLift Servers 遊戲工作階段佇列的最佳實務](#)。

2. 針對 Spot 最佳化佇列建立機群。

根據您的佇列設計，建立機群，將遊戲伺服器部署到所需的位置和執行個體類型。請參閱[建立 Amazon GameLift Servers 受管 EC2 機群](#)協助建立及設定新機群。

3. 建立遊戲工作階段佇列。

新增機群目的地、設定遊戲工作階段置放程序，以及定義置放優先順序。請參閱[建立遊戲工作階段佇列](#)協助建立及設定新佇列。

4. 更新您的遊戲用戶端服務以使用佇列。

當您的遊戲用戶端使用佇列來請求資源時，佇列會避免高中斷機率的資源，並選取符合您定義優先順序的位置。如需協助在遊戲用戶端實作遊戲工作階段配置，請參閱 [建立遊戲工作階段](#)。

5. 更新您的遊戲伺服器以處理 Spot 中斷。

AWS 當 Spot 執行個體需要恢復容量時，可以透過 2 分鐘的通知來中斷 Spot 執行個體。設定您的遊戲伺服器來處理中斷，將玩家影響降至最低。

AWS 在回收 Spot 執行個體之前，它會傳送終止通知。會透過叫用伺服器開發套件回呼函數，將通知 Amazon GameLift Servers 傳遞給所有受影響的 Amazon GameLift Servers 伺服器程序 `onProcessTerminate()`。實作此回呼以結束遊戲工作階段，或將遊戲工作階段和玩家移至新的執行個體。請參閱 [回應伺服器程序關閉通知](#) 協助實作 `onProcessTerminate()`。

Note

AWS 會盡一切努力在回收執行個體之前提供通知，但有可能在警告到達之前 AWS 回收 Spot 執行個體。準備您的遊戲伺服器來處理意外的中斷。

6. 檢閱 Spot 機群和佇列的效能。

在 Amazon GameLift Servers 主控台中或使用 Amazon CloudWatch 檢視 Amazon GameLift Servers 指標，以檢閱效能。如需 Amazon GameLift Servers 指標的詳細資訊，請參閱 [使用 Amazon CloudWatch 監控 Amazon GameLift Servers](#)。關鍵指標包括：

- 中斷率 – 使用 `InstanceInterruptions` 和 `GameSessionInterruptions` 指標來追蹤執行個體和遊戲工作階段的 Spot 相關中斷次數和頻率。回收的遊戲工作階段狀態 AWS 為 `TERMINATED`，狀態原因為 `INTERRUPTED`。
- 佇列有效性 – 追蹤置放成功率、平均等待時間和佇列深度，以確認 Spot 機群不會影響您的佇列效能。
- 機群用量 – 監控執行個體、遊戲工作階段和玩家工作階段的資料。隨需機群的使用量可能是佇列避免置放至 Spot 機群以避免中斷的指標。

Spot 機群佇列的最佳實務

如果您的佇列包含 Spot 機群，請設定彈性佇列。這利用 Spot 機群節省成本，同時將遊戲工作階段中斷的影響降至最低。如需正確建置機群和遊戲工作階段佇列以搭配 Spot 機群使用的協助，請參閱 [教學](#)

課程： [使用 Spot 執行個體建立 Amazon GameLift Servers 佇列](#)。如需 Spot 執行個體的詳細資訊，請參閱 [設計 Spot 執行個體的佇列](#)。

除了上一節的一般最佳實務之外，請考慮這些 Spot 特定的最佳實務：

- 在每個位置建立至少一個隨需機群。隨需機群為您的玩家提供備份遊戲伺服器。您可以保持備份機群縮減規模，直到需要為止，並使用自動擴展功能在 Spot 機群無法使用時增加隨需容量。
- 在某個位置的多個 Spot 機群之間選取不同的執行個體類型。如果某個 Spot 執行個體類型暫時無法使用，中斷只會影響該位置中的一個 Spot 機群。最佳實務是選擇廣泛可用的執行個體類型，並在相同系列中使用執行個體類型（例如 m5.large、m5.xlarge、m5.2xlarge）。使用 [Amazon GameLift Servers 主控台](#) 檢視執行個體類型的歷史定價資料。

設定遊戲工作階段放置的事件通知

您可以使用事件通知來監控個別置放請求的狀態。我們建議為具有大量置放活動的所有遊戲設定事件通知。

設定事件通知有兩種選項。

- 使用佇列將事件通知 Amazon GameLift Servers 發佈至 Amazon Simple Notification Service (Amazon SNS) 主題。
- 使用自動發佈的 Amazon EventBridge 事件及其工具套件來管理事件。

如需發出的遊戲工作階段置放事件清單 Amazon GameLift Servers，請參閱 [遊戲工作階段置放事件](#)。

Important

對於大量置放系統，我們建議使用標準（非 FIFO）Amazon SNS 主題，而不是 FIFO 主題。FIFO 主題的發佈限制低於標準主題，這可能會導致在高負載期間調節例外狀況。如果您使用 FIFO 主題遇到限流，可能會遺失佇列置放通知。

設定 SNS 主題

若要 Amazon GameLift Servers 將遊戲工作階段佇列產生的所有事件發佈至主題，請將通知目標欄位設定為主題。

設定 Amazon GameLift Servers 事件通知用的 SNS 主題

1. 登入 AWS Management Console 並開啟位於 <https://console.aws.amazon.com/sns/v3/home> 的 Amazon SNS 主控台。
2. 在 SNS 主題頁面中，選擇建立主題，然後依照指示建立主題。
3. 在存取政策下，執行下列動作：
 - a. 選擇進階方法。
 - b. 將 JSON 物件的下列粗體區段新增至現有政策。

JSON

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:111122223333:your_topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "your_account"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
```

```

    "Effect": "Allow",
    "Principal": {
      "Service": "gamelift.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-1:111122223333:your_topic_name",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:gamelift:us-east-1:111122223333:gamesessionqueue/your_queue_name"
      }
    }
  }
]
}

```

- c. (選用) 透過將條件新增至資源政策，將其他存取控制新增至主題。
4. 請選擇建立主題。
5. 建立 SNS 主題之後，請在建立佇列期間將其新增至佇列，或編輯現有的佇列以新增該主題。

使用伺服器端加密設定 SNS 主題

使用伺服器端加密 (SSE) 可讓您儲存加密主題中的敏感資料。SSE 使用 AWS Key Management Service (AWS KMS) 中管理的金鑰來保護 Amazon SNS 主題中的訊息內容。如需使用 Amazon SNS 進行伺服器端加密的詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的[靜態加密](#)。

若要使用伺服器端加密設定 SNS 主題，請檢閱下列主題：

- 《AWS Key Management Service 開發人員指南》中的[建立金鑰](#)
- 《Amazon Simple Notification Service 開發人員指南》中的[為主題啟用 SSE](#)

建立 KMS 金鑰時，請使用下列 KMS 金鑰政策：

```

{
  "Effect": "Allow",
  "Principal": {
    "Service": "gamelift.amazonaws.com"
  },
  "Action": [

```

```

        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn":
"arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
"arn:aws:sns:your_region:your_account:your_sns_topic_name"
        }
    }
}

```

設定 EventBridge

Amazon GameLift Servers 會自動將所有遊戲工作階段置放事件發佈至 EventBridge。使用 EventBridge，您可以設定規則，將事件路由到目標進行處理。例如，您可以設定規則，將事件路由 PlacementFulfilled 到 AWS Lambda 函數，處理連線到遊戲工作階段之前的任務。如需 EventBridge 的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 [什麼是 Amazon EventBridge？](#)。

以下是要與 Amazon GameLift Servers 佇列搭配使用的 EventBridge 規則範例：

比對所有 Amazon GameLift Servers 佇列中的事件

```

{
  "source": [
    "aws.gamelift"
  ],
  "detail-type": [
    "GameLift Queue Placement Event"
  ]
}

```

符合來自特定佇列的事件

```

{
  "source": [

```

```
    "aws.gamelift"
  ],
  "detail-type": [
    "GameLift Queue Placement Event"
  ],
  "resources": [
    "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
  ]
}
```

教學課程：使用 Spot 執行個體建立 Amazon GameLift Servers佇列

簡介

本教學課程說明如何為部署在低成本 Spot 機群上的遊戲設定遊戲工作階段放置。Spot 機群需要額外的步驟，以維持玩家的持續遊戲伺服器可用性。

目標對象

本教學課程適用於想要使用 Spot 機群託管自訂遊戲伺服器或的遊戲開發人員Amazon GameLift ServersRealtime。

您將學到什麼

- 定義遊戲工作階段佇列服務的 player 群組。
- 建置機群基礎設施以支援遊戲工作階段佇列的範圍。
- 為每個機群指派別名，以抽象化機群 ID。
- 建立佇列、新增機群，並排定遊戲工作階段Amazon GameLift Servers放置位置的優先順序。
- 新增玩家延遲政策，以協助將延遲問題降至最低。

先決條件

在建立遊戲工作階段置放的機群和佇列之前，請先完成下列任務：

- 檢閱 [Amazon GameLift Servers 的運作方式](#)。
- [整合您的遊戲伺服器與 Amazon GameLift Servers](#)。
- [將您的遊戲伺服器](#)建置或Realtime指令碼上傳至 Amazon GameLift Servers。
- [規劃您的機群組態](#)。

步驟 1：定義佇列的範圍

在本教學課程中，我們為具有一個遊戲伺服器建置變化的遊戲設計佇列。推出時，我們會在兩個位置推出遊戲：亞太區域（首爾）和亞太區域（新加坡）。由於這些位置彼此靠近，延遲不會對我們的玩家造成問題。

在此範例中，有一個玩家客群，這表示我們建立一個佇列。未來，當我們在北美洲發行遊戲時，我們可以建立第二個佇列，其範圍限定為北美玩家。

如需詳細資訊，請參閱[定義佇列的範圍](#)。

步驟 2：建立 Spot 機群基礎設施

使用符合您在 中定義範圍的遊戲伺服器建置或指令碼，在位置 和 中建立機群[步驟 1：定義佇列的範圍](#)。

在本教學課程中，我們建立一個兩個位置基礎設施，每個位置至少有一個 Spot 機群和一個隨需機群。每個機群都會部署相同的遊戲伺服器組建。此外，我們預期在首爾位置的玩家流量會更重，因此我們在那裡新增更多 Spot 機群。

下圖顯示 Spot 機群基礎設施範例，其中 3 個機群位於 ap-northeast-2（首爾）位置，2 個機群位於 ap-southeast-1（新加坡）位置。兩個機群中的所有執行個體都使用建置 MBG_prod_V1。ap-northeast-2 中的機群包含下列機群組態：執行個體類型為 c5.large 的機群 1234_spot_1、執行個體類型為 c5.xlarge 的機群 1234_spot_2，以及執行個體類型為 c5.large 的機群 1234_ondemand。ap-southeast-1 中的機群包含下列機群組態：執行個體類型為 c5.large 的機群 1234_spot_1，以及執行個體類型為 c5.large 的機群 1234_ondemand。

步驟 3：為每個機群指派別名

為基礎設施中的每個機群建立新的別名。別名抽象機群身分，讓定期機群取代更有效率。如需建立別名的詳細資訊，請參閱 [建立 Amazon GameLift Servers 別名](#)。

我們的機群基礎設施有五個機群，因此我們使用路由策略建立五個別名。我們需要亞太區域（首爾）據點的三個別名，以及亞太區域（新加坡）據點的兩個別名。

下圖顯示步驟 2 中所述的 Spot 機群基礎設施，並將別名新增至每個機群。機群 1234_spot_1 具有別名 MBG_spot_1，機群 1234_spot_2 具有別名 MBG_spot_2，機群 1234_ondemand 具有別名 MBG_ondemand。

如需詳細資訊，請參閱[建置多位置佇列](#)。

步驟 4：建立具有目的地的佇列

建立遊戲工作階段佇列並新增您的機群目的地。如需建立佇列的詳細資訊，請參閱[建立遊戲工作階段佇列](#)。

建立佇列時：

- 將預設逾時設定為 10 分鐘。稍後，您可以測試佇列逾時如何影響玩家進入遊戲的等待時間。
- 暫時略過玩家延遲政策的區段。我們將在下一個步驟中說明這一點。
- 排定佇列中機群的優先順序。使用 Spot 機群時，我們建議採用下列其中一種方法：
 - 如果您的基礎設施使用主要位置搭配第二個位置的機群進行備份，請先依位置排定機群的優先順序，再依機群類型排定。
 - 如果您的基礎設施平均使用多個位置，請依機群類型排定機群的優先順序，將 Spot 機群置於佇列頂端。

在本教學課程中，我們會建立名為 `MBG_spot_queue` 的新佇列，並新增所有五個機群的別名。然後，我們會先依位置排定置放的優先順序，再依機群類型排定置放的優先順序。

根據此組態，此佇列一律會嘗試將新的遊戲工作階段放入首爾的 Spot 機群。當這些機群已滿時，佇列會使用首爾隨需機群上的可用容量做為備份。如果全部三個首爾機群都無法使用，會將遊戲工作階段 Amazon GameLift Servers 放置在新加坡機群。

下圖顯示逾時 300 秒並排定目的地優先順序的佇列。目的地的順序如下：ap-northeast-2 中的 1234_spot_1、ap-northeast-2 中的 1234_spot_2、ap-northeast-2 中的 1234_ondemand、ap-southeast-1 中的 1234_spot_1，以及 ap-southeast-1 中的 1234_ondemand。

步驟 5：將延遲限制新增至佇列

我們的遊戲在遊戲工作階段置放請求中包含延遲資訊。我們也有玩家隊伍功能，可為一組玩家建立遊戲工作階段。我們可以讓玩家等待更長的時間，才能以理想的遊戲體驗進入遊戲。我們的遊戲測試顯示下列觀察：

- 50 毫秒以下的延遲是理想的。
- 遊戲無法以超過 250 毫秒的延遲播放。
- 玩家在大約一分鐘時變得不耐心。

對於我們的佇列，如果逾時 300 秒，我們會新增政策陳述式來限制允許的延遲。政策陳述式會逐漸允許高達 250 毫秒延遲的較大延遲值。

使用此政策，我們的佇列會在第一分鐘尋找具有理想延遲（低於 50 毫秒）的位置，然後放寬限制。佇列不會放置玩家延遲為 250 毫秒或更高的位置。

下圖顯示步驟四新增玩家延遲政策的佇列。玩家延遲政策狀態、強制執行 60 秒的 50 毫秒限制、強制執行 30 秒的 125 毫秒限制，以及強制執行 250 毫秒限制，直到逾時為止。

Summary

恭喜您！以下是您完成的事項：

- 您有一個遊戲工作階段佇列，範圍限定於玩家群體的區段。
- 您的佇列可有效使用 Spot 機群，並在 Spot 中斷發生時具有彈性。
- 您的佇列會優先考慮機群，以獲得最佳玩家體驗。
- 佇列具有延遲限制，可保護玩家免於遊戲體驗不佳。

您現在可以使用佇列來為其服務的玩家放置遊戲工作階段。為這些玩家提出遊戲工作階段置放請求時，請在請求中參考此遊戲工作階段佇列名稱。如需提出遊戲工作階段置放請求的詳細資訊，請參閱 [建立遊戲工作階段](#) 或 [整合的遊戲用戶端 Amazon GameLift ServersRealtime](#)。

後續步驟：

- [設計您自己的佇列](#)。
- [建立佇列](#)。
- [將佇列與遊戲用戶端搭配使用](#)。

使用 擴展遊戲託管容量 Amazon GameLift Servers

在執行個體中測量的託管容量代表Amazon GameLift Servers可同時託管的遊戲工作階段數量，以及這些遊戲工作階段可容納的並行玩家數量。遊戲託管最具挑戰性的任務之一是擴展容量以滿足玩家需求，而不會浪費在您不需要的資源上。如需詳細資訊，請參閱[擴展機群容量的規模](#)。

容量會在機群位置層級進行調整。所有機群至少有一個位置：機群的主 AWS 區域。檢視或擴展容量時，資訊會依位置列出，包括機群的主區域和任何其他遠端位置。

您可以手動設定要維護的執行個體數量，也可以設定自動擴展，以在玩家需求變更時動態調整容量。建議您先開啟目標型自動擴展選項。以目標為基礎的自動擴展目標是維護足夠的託管資源，以容納目前的玩家，再加上一些額外的資源來處理玩家需求的意外尖峰。對於大多數遊戲，目標型自動擴展提供高效率的擴展解決方案。

您可以使用 Amazon GameLift Servers主控台執行大多數機群擴展活動。您也可以使用 AWS SDK 或 AWS Command Line Interface (AWS CLI) 搭配 [的服務 API Amazon GameLift Servers](#)。

主題

- [在 主控台中管理機群容量](#)
- [設定Amazon GameLift Servers容量限制](#)
- [手動設定Amazon GameLift Servers機群的容量](#)
- [使用 Amazon GameLift Servers 自動調整機群容量的規模](#)
- [擴展Amazon GameLift Servers容器機群](#)

在 主控台中管理機群容量

1. 開啟 [Amazon GameLift Servers 主控台](#)。
2. 在導覽窗格中，選擇託管、機群。
3. 在機群頁面上，選擇作用中機群的名稱，以開啟機群的詳細資訊頁面。
4. 選擇擴展索引標籤。在此索引標籤上，您可以：
 - 檢視整個機群的歷史擴展指標。
 - 檢視和更新每個機群位置的容量設定，包括擴展限制和目前的容量設定。
 - 更新目標型自動擴展、檢視套用至整個機群的規則型自動擴展政策，以及暫停每個位置的自動擴展活動。

設定Amazon GameLift Servers容量限制

手動或透過自動擴展擴展機Amazon GameLift Servers群位置的託管容量時，請考慮該位置的擴展限制。所有機群位置都有定義位置容量允許範圍的下限和上限。根據預設，機群位置的限制最少有 0 個執行個體，最多有 1 個執行個體。在擴展機群位置之前，請調整限制。

如果您使用的是自動擴展，則最大限制Amazon GameLift Servers允許擴展機群位置以滿足玩家需求，但可防止流失託管成本，例如在 DDOS 攻擊期間。設定 [Amazon CloudWatch 警示](#) 以在容量接近上限時通知您，以便您可以評估情況並視需要手動調整。（您也可以[建立帳單警示](#)來監控 AWS 成本。）即使玩家需求較低，最低限制也有助於維持託管可用性。

您可以在 [Amazon GameLift Servers主控台](#) 或使用 AWS Command Line Interface () 來設定機群位置的容量限制AWS CLI。

設定容量限制

Console

1. 開啟 [Amazon GameLift Servers 主控台](#)。
2. 在導覽窗格中，選擇託管、機群。
3. 在機群頁面上，選擇作用中機群的名稱，以開啟機群的詳細資訊頁面。
4. 在擴展索引標籤的擴展容量下，選取機群位置，然後選擇編輯。
5. 在編輯擴展容量對話方塊中，設定大小下限、所需執行個體和大小上限的執行個體計數。
6. 選擇確認。

AWS CLI

1. 檢查目前的容量設定。在命令列視窗中，使用 [describe-fleet-location-capacity](#) 命令搭配您要變更容量的機群 ID 和位置。此命令會傳回包含位置目前容量設定的 [FleetCapacity](#) 物件。判斷新的執行個體限制是否可以容納目前所需的執行個體設定。

```
aws gamelift describe-fleet-location-capacity \
  --fleet-id <fleet identifier> \
  --location <location name>
```

2. 更新限制設定。在命令列視窗中，使用 [update-fleet-capacity](#) 命令搭配下列的參數。您可以使用同一個命令，來調整執行個體限制和所要的執行個體數量。

```
--fleet-id <fleet identifier>
--location <location name>
--max-size <maximum capacity for scaling>
--min-size <minimum capacity for scaling>
--desired-instances <fleet capacity goal>
```

範例：

```
aws gamelift update-fleet-capacity \
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --location us-west-2 \
  --max-size 10 \
  --min-size 1 \
  --desired-instances 10
```

如果您的請求成功，會Amazon GameLift Servers傳回機群 ID。如果新的 max-size或 min-size值與目前的desired-instances設定衝突，會Amazon GameLift Servers傳回錯誤。

手動設定Amazon GameLift Servers機群的容量

當您建立新的機群時，Amazon GameLift Servers會自動將所需的執行個體設定為每個機群位置中的一個執行個體。然後，Amazon GameLift Servers 在每個位置部署一個新執行個體。若要變更機群容量，您可以新增目標型自動擴展政策，也可以手動設定要用於位置的執行個體數量。如需詳細資訊，請參閱[擴展機群容量的規模](#)。

當您不需要自動擴展或需要將容量保留在指定層級時，手動設定機群的容量會很有用。只有在您未使用目標型自動擴展政策時，手動設定容量才有效。如果您有目標型自動擴展政策，它會根據自己的擴展規則立即重設所需的容量。

您可以在 Amazon GameLift Servers主控台或使用 AWS Command Line Interface () 手動設定容量 AWS CLI。機群的狀態必須為作用中。

暫停自動擴展

您可以暫停每個機群位置的所有自動擴展活動。在自動擴展暫停的情況下，除非手動變更，否則機群位置中所需的執行個體數量會保持不變。當您暫停位置的自動擴展時，會影響機群的目前政策和您未來可能定義的任何政策。

手動設定機群容量

Console

1. 開啟 [Amazon GameLift Servers 主控台](#)。
2. 在導覽窗格中，選擇託管、機群。
3. 在機群頁面上，選擇作用中機群的名稱，以開啟機群的詳細資訊頁面。
4. 在擴展索引標籤的暫停自動擴展位置下，選取要暫停自動擴展的每個位置，然後選擇暫停。
5. 在擴展容量下，選取要手動設定的位置，然後選擇編輯。
6. 在編輯擴展容量對話方塊中，設定所需執行個體的偏好值，然後選擇確認。這會告知要維持在作用中狀態Amazon GameLift Servers的執行個體數量，並準備好託管遊戲工作階段。

Amazon GameLift Servers 透過部署其他執行個體或關閉不需要的執行個體來回應變更。當 Amazon GameLift Servers 完成此程序時，位置中的作用中執行個體數目會變更，以符合更新的所需執行個體值。此程序可能需要一些時間。

AWS CLI

1. 檢查目前的容量設定。在命令列視窗中，使用 [describe-fleet-location-capacity](#) 命令搭配您要變更容量的機群 ID 和位置。此命令會傳回包含位置目前容量設定的 [FleetCapacity](#) 物件。判斷執行個體限制是否可以容納新的所需執行個體設定。

```
aws gamelift describe-fleet-location-capacity \
  --fleet-id <fleet identifier> \
  --location <location name>
```

2. 更新所需的容量。使用 [update-fleet-capacity](#) 命令搭配所需執行個體的機群 ID、位置和新值。如果此值超出目前的限制範圍，您可以在相同的命令中調整限制值。

```
--fleet-id <fleet identifier>
--location <location name>
--desired-instances <fleet capacity as an integer>
--max-size <maximum capacity> [Optional]
--min-size <minimum capacity> [Optional]
```

範例：

```
aws gamelift update-fleet-capacity \
```



```
--fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
--location us-west-2 \  
--desired-instances 5 \  
--max-size 10 \  
--min-size 1
```

如果您的請求成功，會Amazon GameLift Servers傳回機群 ID。如果新的所需執行個體設定超出下限和上限，Amazon GameLift Servers會傳回錯誤。

使用 Amazon GameLift Servers 自動調整機群容量的規模

使用 中的自動擴展Amazon GameLift Servers功能，動態擴展機群容量以回應遊戲伺服器活動。隨著玩家抵達和開始遊戲工作階段，自動擴展可以新增更多執行個體；隨著玩家需求減少，自動擴展可以終止不需要的執行個體。自動擴展是將託管資源和成本降至最低的有效方式，同時仍提供順暢、快速的玩家體驗。

若要使用自動擴展，您可以建立擴展政策，告知Amazon GameLift Servers何時擴展或縮減。擴展政策有兩種類型：目標型和規則型。以目標為基礎的方法 - 目標追蹤 - 是完整的解決方案。我們建議您將其作為最簡單且最有效的選項。規則型擴展政策需要您定義自動擴展決策程序的每個層面，這對於解決特定問題非常有用。此解決方案最適合做為目標型自動擴展的補充。

您可以使用 Amazon GameLift Servers主控台、AWS Command Line Interface (AWS CLI) 或 AWS SDK 來管理目標型自動擴展。您只能使用 AWS CLI 或 AWS SDK 管理規則型自動擴展，但您可以在主控台中檢視規則型擴展政策。

主題

- [目標型自動擴展](#)
- [使用規則型政策自動擴展](#)

目標型自動擴展

的目標型自動擴展會根據機群指標 Amazon GameLift Servers調整容量層級PercentAvailableGameSessions。此指標代表機群的可用緩衝區，用於玩家需求突然增加。

維持容量緩衝的主要原因，是玩家的等待時間。當遊戲工作階段槽準備就緒並等待時，讓新玩家進入遊戲工作階段需要幾秒鐘的時間。如果沒有可用的資源，玩家必須等到現有的遊戲工作階段結束，或是等到有新的可用資源。啟動新的執行個體和伺服器程序可能需要幾分鐘的時間。

設定目標型自動擴展時，請指定您希望機群維護的緩衝區大小。由於會PercentAvailableGameSessions測量可用資源的百分比，因此實際緩衝區大小是機群總容量的百分比。Amazon GameLift Servers 新增或移除執行個體以維持目標緩衝區大小。使用大型緩衝區，您可以將等待時間降至最低，但您還需要支付可能無法使用的額外資源。如果玩家較能容忍等待時間，您可以設定較小的緩衝容量來降低成本。

設定目標型自動擴展

Console

1. 開啟 [Amazon GameLift Servers 主控台](#)。
2. 在導覽窗格中，選擇託管、機群。
3. 在機群頁面上，選擇作用中機群的名稱，以開啟機群的詳細資訊頁面。
4. 選擇擴展索引標籤。此標籤會顯示機群的規模調整歷史指標，並且包含控制項，可用來調整目前的規模調整設定。
5. 在擴展容量下，檢查最小大小和最大大小限制是否適合機群。啟用自動擴展後，容量會在這兩個限制之間調整。
6. 在目標型自動調整規模政策中，選擇編輯。
7. 在編輯目標型自動擴展政策對話方塊中，針對可用遊戲工作階段的百分比，設定您要維護的百分比，然後選擇確認。在您確認設定之後，會在目標型自動擴展政策下Amazon GameLift Servers新增以目標為基礎的政策。

AWS CLI

1. 設定容量限制。使用 [update-fleet-capacity](#) 命令設定限制值。如需詳細資訊，請參閱[設定 Amazon GameLift Servers 容量限制](#)。
2. 建立新政策。開啟命令列視窗，然後使用 [put-scaling-policy](#) 命令搭配政策的參數設定。若要更新現有的政策，請指定政策的名稱，並提供已更新政策的完整版本。

```
--fleet-id <unique fleet identifier>
--name "<unique policy name>"
--policy-type <target- or rule-based policy>
--metric-name <name of metric>
--target-configuration <buffer size>
```

範例：

```
aws gamelift put-scaling-policy \  
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \  
  --name "My_Target_Policy_1" \  
  --policy-type "TargetBased" \  
  --metric-name "PercentAvailableGameSessions" \  
  --target-configuration "TargetValue=5"
```

使用規則型政策自動擴展

中的規則型擴展政策可在自動擴展機群容量以回應玩家活動時Amazon GameLift Servers提供精細的控制。對於每個政策，您可以將擴展連結至多個機群指標之一、識別觸發點，以及自訂回應擴展或縮減事件。規則型政策適用於補充[目標型擴展](#)，以處理特殊情況。

規則型政策陳述以下內容：「如果機群指標符合或超過閾值達特定時間長度，則將機群的容量變更為指定的數量。」本主題介紹用於建構政策語句的語法，並提供說明以建立和管理以規則為基礎的政策。

管理以規則為基礎的政策

使用 AWS SDK 或 AWS Command Line Interface (AWS CLI) 搭配 [的服務 API Amazon GameLift Servers](#) 來建立、更新或刪除規則型政策。您可以在 Amazon GameLift Servers 主控台中查看所有使用中的政策。

若要暫時停止機群的所有擴展政策，請使用 AWS CLI 命令 [stop-fleet-actions](#)。

若要建立或更新規則型擴展政策 (AWS CLI)：

1. 設定容量限制。使用 [update-fleet-capacity](#) 命令來設定其中一項或同時設定兩項限制值。如需詳細資訊，請參閱[設定Amazon GameLift Servers容量限制](#)。
2. 建立新政策。開啟命令列視窗，並使用 [put-scaling-policy](#) 命令搭配政策的參數設定。若要更新現有的政策，請指定政策的名稱，並提供已更新政策的完整版本。

```
--fleet-id <unique fleet identifier>  
--name "<unique policy name>"  
--policy-type <target- or rule-based policy>  
--metric-name <name of metric>  
--comparison-operator <comparison operator>  
--threshold <threshold integer value>  
--evaluation-periods <number of minutes>  
--scaling-adjustment-type <adjustment type>
```

```
--scaling-adjustment <adjustment amount>
```

範例：

```
aws gamelift put-scaling-policy \  
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --name "Scale up when AGS<50" \  
  --policy-type RuleBased \  
  --metric-name AvailableGameSessions \  
  --comparison-operator LessThanThreshold \  
  --threshold 50 \  
  --evaluation-periods 10 \  
  --scaling-adjustment-type ChangeInCapacity \  
  --scaling-adjustment 1
```

若要使用 刪除規則型擴展政策 AWS CLI：

- 開啟命令列視窗，並使用 [delete-scaling-policy](#) 命令搭配機群 ID 和政策名稱。

範例：

```
aws gamelift delete-scaling-policy \  
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --name "Scale up when AGS<50"
```

自動擴展規則的語法

若要建構規則型擴展政策陳述式，請指定六個變數：

如果 *<metric name>* 保持為 *<comparison operator>* *<threshold value>* 達 *<evaluation period>*，則使用 *<adjustment type>* 將機群容量更改為 *<adjustment value>*，或者按後者幅度更改。

例如，每當機群的額外容量低於處理 50 個新遊戲工作階段所需的容量時，此政策陳述式就會啟動擴展事件：

如果 AvailableGameSessions 維持在 less than 50 達 10 minutes，那麼請使用 ChangeInCapacity 依照 1 instances 的幅度來更改機群容量。

指標名稱

若要啟動擴展事件，請將自動擴展政策連結至下列其中一個機群特定的指標。如需完整的指標說明，請參閱 [機群的 Amazon GameLift Servers 指標](#)。

- 啟動遊戲工作階段
- 使用中的遊戲工作階段
- 可用的遊戲工作階段
- 可用遊戲工作階段所占百分比
- 使用中的執行個體
- 可用的玩家工作階段
- 目前玩家工作階段
- 閒置執行個體
- 閒置執行個體所占百分比

如果機群位於遊戲工作階段佇列中，您可以使用下列指標：

- 佇列深度 – 此機群的最佳可用託管位置是待處理遊戲工作階段請求的數量。
- 等待時間 – 機群特定的等待時間。最早的待處理遊戲工作階段請求已等待執行的時間長度。機群的等待時間等於佇列中目前最早請求的時間。

比較運算子

Amazon GameLift Servers 說明如何比較指標資料與閾值。有效的比較運算子包括大於 (>)、小於 (<)、大於或等於 (>=)，以及小於或等於 (<=)。

閾值

當指定的指標值符合或超過閾值時，就會啟動擴展事件。此值一律為正整數。

評估期間

在開始擴展事件之前，指標必須符合或超過評估期間完整長度的閾值。評估期長度是連續的；如果指標退回到閾值之下，則評估期重新開始。

調整類型和值

這組變數可一起運作，指定擴展事件開始時 Amazon GameLift Servers 應如何調整機群的容量。從三種可能的調整類型中選擇：

- 容量變更 – 增加或減少指定執行個體數量的目前容量。將調整值設為要在機群中增加或刪除的執行個體數。正值會新增執行個體，而負值會刪除執行個體。例如，「-10」的值會將機群縮減 10 個執行個體，無論機群的總大小為何。

- 容量的百分比變更 – 將目前容量增加或減少指定的百分比。將調整值設定為您要增加或減少機群容量的百分比。正值會新增執行個體，而負值會刪除執行個體。例如，對於具有 50 個執行個體的機群，「20」的百分比變更會將 10 個執行個體新增至機群。
- 精確容量 – 將目前容量增加或減少為特定值。將調整值設為您希望在機群中維持的精確執行個體數。

規則型自動擴展的提示

下列建議可協助您使用規則型政策充分利用自動擴展。

使用多個政策

您可以同時有多個機群的自動擴展政策。最常見的情況是：使用一個以目標為基礎的政策來管理大多數擴展需求；而使用以規則為基礎的政策來處理邊緣情況。使用多個政策沒有限制。

使用多個政策時，每個政策都會獨立運作。無法控制擴展事件的序列。例如，如果您有多個政策正在向上擴展，則玩家活動可能會同時啟動多個擴展事件。避免互相啟動的政策。例如，如果您建立的擴展和縮減政策將容量設定為超過彼此的閾值，您可以建立無限迴圈。

設定最大和最小容量

每個機群都具有最大和最小容量限制。使用自動擴展時，此功能很重要。自動擴展永遠不會將容量設定為超出此範圍的值。預設情況下，新建立的機群具有最小值 0 和最大值 1。若要讓自動擴展政策如預期影響容量，請增加最大值。

機群容量也會受限於機群的執行個體類型限制，以及 中的服務配額 AWS 帳戶。您無法在這些限制和帳戶配額之外設定下限和上限。

在容量更改後追蹤指標

變更容量以回應自動擴展政策後，會Amazon GameLift Servers等待 10 分鐘，然後再回應來自相同政策的觸發。此等待可讓您有Amazon GameLift Servers時間新增執行個體、啟動遊戲伺服器、連接玩家，以及開始從新執行個體收集資料。在此期間，會根據 指標Amazon GameLift Servers評估政策，並追蹤政策的評估期間，該期間會在發生擴展事件後重新啟動。這表示擴展政策可能會在等待時間結束後立即啟動另一個擴展事件。

不同自動擴展政策開始的擴展事件之間沒有等待時間。

擴展Amazon GameLift Servers容器機群

遊戲託管最具挑戰性的任務之一是擴展容量以滿足玩家需求，而不會浪費在您不需要的資源上。在受管容器機群中，您可以透過新增或移除機群執行個體來擴展機群容量。

當您建立新的機群時，Amazon GameLift Servers會將機群所需的容量設定為一個執行個體，並在機群的主區域中部署一個執行個體。對於多位置機群，Amazon GameLift Servers 會將一個執行個體部署到主要區域和每個遠端位置。機群狀態達到 後ACTIVE，您可以提高或降低所需的容量以縮減規模。

您可以使用Amazon GameLift Servers擴展功能來手動變更容量，或根據玩家需求設定自動擴展：

- 使用目標追蹤設定自動擴展。請參閱 [目標型自動擴展](#)。
- 手動變更機群的容量。請參閱 [手動設定Amazon GameLift Servers機群的容量](#)。

擴展容器機群時，請考慮新增或移除執行個體如何影響機群託管遊戲工作階段和玩家的容量。

- 每個執行個體的遊戲工作階段
 - 在執行個體上執行的每個遊戲伺服器程序代表託管一個遊戲工作階段的容量。
 - 使用此公式來計算在容器機群執行個體上同時執行的遊戲工作階段數量：

```
[Game sessions per instance] = [# of game server processes per game server container] * [# of game server container groups per instance]
```

如果您的容器架構在遊戲伺服器容器中同時執行一個遊戲伺服器程序，則每個執行個體的遊戲工作階段等於每個執行個體的遊戲伺服器容器群組數量。

- 對於每個執行個體的遊戲伺服器容器群組，請呼叫 [DescribeContainerFleet](#) 以取得 `GameServerContainerGroupsPerInstance` 或 `MaximumGameServerContainerGroupsPerInstance` 值。
- 每個執行個體的玩家數
 - 您決定在每個遊戲工作階段中允許的玩家位置數量。根據託管解決方案處理遊戲工作階段置放的方式，您可以在配對組態或呼叫中為每個遊戲工作階段定義玩家，以開始遊戲工作階段置放。
 - 使用此公式來計算可在容器機群執行個體上同時玩遊戲的玩家數量：

```
[Players per instance] = [# of game sessions per instance] * [# of player slots per game session]
```


若要取得容器機群目前的總容量，請呼叫 [DescribeFleetCapacity](#) 或 [DescribeFleetLocation](#) 容量，以取得機群中的遊戲伺服器容器群組數量。作用中群組是目前託管遊戲工作階段的群組。閒置群組已準備好託管新的遊戲工作階段。將這些值乘以每個遊戲伺服器容器群組的伺服器程序數目。

準備您的遊戲以使用 Amazon GameLift Servers 託管啟動

使用下列檢查清單來驗證遊戲的每個部署階段。標記為 **【關鍵】** 的項目對您的生產啟動至關重要。

下載並完成 Amazon GameLift Servers 啟動問卷，此問卷可在 [Amazon GameLift Servers 主控台](#) 取得。我們希望每個使用 Amazon GameLift Servers 的遊戲開發人員都能擁有順暢的啟動日，而請求的資訊有助於我們準備即將到來的負載測試、軟啟動或公開啟動。計劃在執行第一次載入測試之前至少三 (3) 個月提交完成的問卷。

主題

- [準備好您的遊戲](#)
- [準備進行測試](#)
- [準備啟動](#)
- [規劃啟動後更新](#)

準備好您的遊戲

- **【關鍵】** 確認您已完成託管解決方案的所有 [開發藍圖步驟](#)，而且您已備妥所有必要的元件，包括整合的遊戲伺服器、遊戲用戶端的後端服務、託管機群，以及遊戲工作階段放置方法（例如佇列）。
- **【關鍵】** [Create AWS Identity and Access Management \(IAM\) 角色](#)，可讓您的遊戲伺服器在執行時存取其他 AWS 資源。
- **【關鍵】** 視需要設計和實作容錯移轉至其他託管資源。
- 考慮遊戲的佇列和 [機群結構](#)，[規劃將機群推展到目標位置](#)。
- 透過 AWS CloudFormation 和 使用基礎設施即程式碼 (IaC) [自動化部署](#) AWS Cloud Development Kit (AWS CDK)。
- 使用 Amazon CloudWatch 和 Amazon Simple Storage Service (Amazon S3) [收集日誌和分析](#)。

準備進行測試

- **【關鍵】** [請求增加 Amazon GameLift Servers 服務配額](#) 和其他 AWS 服務 配額，以便您的即時環境可以根據生產需求進行擴展。
- **【關鍵】** 驗證即時機群上的開放連接埠是否符合伺服器可以使用的連接埠範圍。
- **【關鍵】** 關閉 RDP 連接埠 3389 和 SSH 連接埠 22。

- 制定遊戲的 DevOps 管理計劃。如果您使用 Amazon CloudWatch Logs 或 Amazon CloudWatch 自訂指標，請定義伺服器機群上嚴重或重大問題的警示。模擬失敗並測試 Runbook。
- 確認您使用的運算資源可支援您要在每個運算上同時執行的伺服器程序數目。
- 先[調整您的擴展政策](#)，使其更為保守，並提供比您想像中更多的閒置容量。您可以稍後針對成本進行最佳化。考慮使用閒置容量為 20% 的目標型擴展政策。
- 對於 FlexMatch，請使用[延遲規則](#)來比對地理位置接近彼此的玩家。使用來自負載測試用戶端的合成延遲資料，測試其在負載下的行為。
- 載入測試您的玩家身分驗證和遊戲工作階段基礎設施，以查看它是否有效擴展以滿足需求。
- 確認持續執行數天的伺服器仍可接受連線。
- 將您的 支援 計劃層級提升為商業或企業，讓 AWS 可以在問題或中斷期間回應您。

準備啟動

- **【關鍵】** [將機群保護政策設定為](#)對所有即時機群提供完整保護，以便縮減規模不會停止作用中的遊戲工作階段。
- **【關鍵】** [將機群大小上限](#)設定為足夠高，以適應最高峰的預期需求。我們建議您針對非預期的需求將大小上限加倍。
- 鼓勵您的整個開發團隊參與啟動事件，並監控您在啟動室中的遊戲啟動。
- 監控玩家延遲和玩家體驗。

規劃啟動後更新

- [調校擴展政策](#)，以根據玩家用量將閒置容量降至最低。
- 根據玩家延遲資料和修訂後的需求[修改FlexMatch規則](#)或[新增託管位置](#)。
- 最佳化執行時間組態，以盡可能在每個運算資源上執行任意數量的遊戲工作階段。以這種方式最佳化效能效率可能會直接影響機群成本，因為您可能可以使用相同的運算資源執行更多伺服器程序。
- [使用您的分析資料](#)來推動持續開發、改善玩家體驗和遊戲壽命，以及最佳化獲利。

使用 管理Amazon GameLift Servers託管資源 AWS CloudFormation

您可以使用 AWS CloudFormation 來管理您的 Amazon GameLift Servers 資源。在 AWS CloudFormation 中，您可以建立範本，以建立每個資源的模型，然後使用範本來建立您的資源。若要更新資源，您可以變更範本，並使用 AWS CloudFormation 實作更新。您可以將資源組織成邏輯群組，稱為堆疊和堆疊集。

使用 AWS CloudFormation 維護您的 Amazon GameLift Servers 託管資源，可讓您更有效率地管理 AWS 資源集。您可以使用版本控制來追蹤一段時間內的範本變更，並協調多個專案團隊成員所做的更新。您也可以重複使用範本。例如，跨多個區域部署遊戲時，您可以使用相同的範本，在每個區域建立相同的資源。您也可以使用這些範本，在另一個分割區中部署相同的資源集。

如需的詳細資訊 AWS CloudFormation，請參閱[AWS CloudFormation 《使用者指南》](#)。若要檢視 Amazon GameLift Servers 資源的範本資訊，請參閱[Amazon GameLift Servers 資源類型參考](#)。

最佳實務

如需使用的詳細指引 AWS CloudFormation，請參閱AWS CloudFormation 《使用者指南》中的[AWS CloudFormation 最佳實務](#)。此外，這些最佳實務與 Amazon GameLift Servers 具有特殊關係。

- 透過 持續管理您的 資源 AWS CloudFormation。如果您在資源之外變更資源 AWS CloudFormation，將會與資源範本不同步。
- 使用 AWS CloudFormation 堆疊和堆疊集來有效管理多個資源。
 - 使用堆疊來管理連線資源群組。例如，包含組建的堆疊、參考組建的機群，以及參考機群的別名。如果您更新範本以取代組建，會 AWS CloudFormation 取代連線至組建的機群。AWS CloudFormation 然後，會更新現有的別名以指向新的機群。如需詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[使用堆疊](#)。
 - 如果您要跨多個區域或 AWS 帳戶部署相同的 AWS CloudFormation 堆疊，請使用堆疊集。如需詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[使用堆疊集](#)。
- 如果您使用的是 Spot 執行個體，請包含隨需機群做為備份。我們建議您在每個區域中設定具有兩個機群的範本、一個具有 Spot 執行個體的機群，以及一個具有隨需執行個體的機群。
- 當您管理多個位置的資源時，請將特定位置的資源和全域資源分組到不同的堆疊。
- 將全域資源放在靠近使用它的 服務附近。佇列和配對組態等資源往往會收到來自特定來源的大量請求。透過將資源放在這些請求的來源附近，您可以將請求行程時間降至最低，並提高整體效能。

- 將您的配對組態放置在與其使用之遊戲工作階段佇列的相同區域。
- 為堆疊中的每個機群建立不同別名。

使用 AWS CloudFormation 堆疊

建議您在設定 Amazon GameLift Servers 資源的 AWS CloudFormation 堆疊時，使用下列結構。您的最佳堆疊結構會因您要將遊戲部署在一個位置或多個位置而有所不同。

單一位置的堆疊

若要在單一位置管理 Amazon GameLift Servers 資源，我們建議使用兩堆疊結構：

- 支援堆疊 – 此堆疊包含資源所依賴 Amazon GameLift Servers 的資源。至少，此堆疊應該包含 S3 儲存貯體，其中儲存自訂遊戲伺服器或 Realtime 指令碼檔案。堆疊也應包含 IAM 角色，該角色在建立 Amazon GameLift Servers 組建或指令碼資源時授予從 S3 儲存貯體擷取檔案的 Amazon GameLift Servers 許可。此堆疊也可能包含遊戲中使用的其他 AWS 資源，例如 DynamoDB 資料表、Amazon Redshift 叢集和 Lambda 函數。
- Amazon GameLift Servers 堆疊 – 此堆疊包含您的所有 Amazon GameLift Servers 資源，包括組建或指令碼、一組機群、別名和遊戲工作階段佇列。會使用存放在 S3 儲存貯體位置中的檔案 AWS CloudFormation 建立組建或指令碼資源，並將組建或指令碼部署至一或多個機群資源。每個機群都應該有一個對應別名。遊戲工作階段佇列會參考部分或全部機群別名。如果您是使用 FlexMatch 進行配對，此堆疊也會包含配對組態和規則集。

下圖說明在單一 AWS 區域中部署資源的雙堆疊結構。

多個區域的堆疊

在多個區域中部署遊戲時，請記住資源如何跨區域互動。某些資源 (例如 Amazon GameLift Servers 機群) 只能參考同一區域中的其他資源。其他資源 (例如 Amazon GameLift Servers 佇列) 與區域無關。若要管理多個區域中的 Amazon GameLift Servers 資源，我們建議使用下列結構。

- 區域支援堆疊 – 這些堆疊包含資源所依賴 Amazon GameLift Servers 的資源。此堆疊必須包括 S3 儲存貯體，其中儲存自訂遊戲伺服器或 Realtime 指令碼檔案。它也可能包含遊戲的其他 AWS 資源，例如 DynamoDB 資料表、Amazon Redshift 叢集和 Lambda 函數。其中許多資源都是區域特定的，因此您必須在每個區域中建立這些資源。Amazon GameLift Servers 也需要允許存取這些支援資源的

IAM 角色。由於 IAM 角色與區域無關，因此您只需要一個角色資源，放在任何區域中，並在所有其他支援堆疊中參考。

- 區域Amazon GameLift Servers堆疊 – 此堆疊包含部署遊戲的每個區域中必須存在Amazon GameLift Servers的資源，包括組建或指令碼、一組機群和別名。AWS CloudFormation 會建立組建或指令碼資源，並在 S3 儲存貯體位置中使用檔案，並將組建或指令碼部署至一或多個機群資源。每個機群都應該有一個對應別名。遊戲工作階段佇列會參考部分或全部機群別名。您可以維護一個範本，來描述這種類型的堆疊，並使用它在每個區域建立相同的資源集。
- 全域Amazon GameLift Servers堆疊 – 此堆疊包含您的遊戲工作階段佇列和配對資源。這些資源可以位於任何區域，且通常放置在相同區域中。佇列可以參考位於任何區域中的機群或別名。若要在不同的區域中放置其他佇列，請建立額外的全域堆疊。

下圖說明在數個 AWS 區域中部署資源的多堆疊結構。第一個圖表顯示單一遊戲工作階段佇列的結構。第二個圖表顯示具有多個佇列的結構。

更新組建

Amazon GameLift Servers 建置是不可變的，就像建置與機群之間的關係一樣。因此，當您更新託管資源，以使用一組新的遊戲建置檔案時，必須發生下列情況：

- 使用一組新檔案來建立新建置 (取代)。
- 建立一組新的機群來部署新的遊戲建置 (取代)。
- 重新導向別名以指向新的機群 (更新而不中斷)。

如需詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[更新堆疊資源的行為](#)。

自動部署建置更新

更新包含相關組建、機群和別名資源的堆疊時，預設 AWS CloudFormation 行為是依序自動執行這些步驟。您可以先將新的建置檔案上傳到新的 S3 位置，以觸發此更新。然後，您修改 AWS CloudFormation 建置範本以指向新的 S3 位置。當您使用新的 S3 位置更新堆疊時，這會觸發以下 AWS CloudFormation 順序：

1. 從 S3 擷取新檔案、驗證這些檔案，並建立新的 Amazon GameLift Servers 建置。
2. 更新機群範本中的建置參考，這會觸發新的機群建立。

3. 在新的機群作用中之後，更新別名中的機群參考，這會觸發別名更新，以新機群做為目標。
4. 刪除舊機群。
5. 刪除舊建置。

如果您的遊戲工作階段佇列使用機群別名，玩家流量會在別名更新後立即自動切換至新機群。隨著遊戲工作階段結束，舊機群會逐漸耗盡玩家。自動調整規模可處理在玩家流量變動時，新增和移除每組機群中執行個體的任務。或者，您可以指定初始所需的執行個體計數，以快速提升交換，並在稍後啟用自動調整規模。

您也可以 AWS CloudFormation 保留資源，而不是刪除資源。如需詳細資訊，請參閱《AWS CloudFormation API 參考》中的 [RetainResources](#)。

手動部署建置更新

如果想要更好地控制新機群何時上線，供玩家使用，您有一些選項可以選擇。您可以選擇使用 Amazon GameLift Servers 主控台或 CLI 手動管理別名。或者，您可以將第二組建置和機群定義新增至範本，而不是更新您的建置範本以取代建置和機群。當您更新範本時，會 AWS CloudFormation 建立第二個建置資源和對應的機群。由於未取代現有的資源，因此不會刪除它們，而且別名仍會指向原始機群。

使用這種方法的主要優點是它為您提供了靈活性。您可以為新版建置建立個別資源、測試新資源，以及控制新機群何時上線，供玩家使用。潛在的缺點是，短時間內它需要在每個區域有兩倍多的資源。

下圖說明此程序。

轉返的運作方式

執行資源更新時，若有任何步驟未順利完成，則 AWS CloudFormation 會自動啟動轉返。此程序會依序反轉每個步驟，同時刪除新建立的資源。

如果您需要手動觸發轉返，請將建置範本的 S3 位置索引鍵變回原始位置，並更新您的堆疊。這時會建立新的 Amazon GameLift Servers 建置和機群，而且在機群作用中之後，別名會切換至新機群。如果您是個別管理別名，則需要切換它們以指向新機群。

如需如何處理失敗或卡住的轉返的詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[繼續轉返更新](#)。

監控 Amazon GameLift Servers

如果您使用 Amazon GameLift Servers FleetIQ 作為 Amazon EC2 的獨立功能，請參閱 [《Amazon EC2 使用者指南》](#) 中的 [Amazon EC2 中的安全性](#)。Amazon EC2

監控是維護 Amazon GameLift Servers 和其他 AWS 解決方案的可靠性、可用性和效能的重要部分。Amazon GameLift Servers 的指標有三種主要用途：監控系統運作狀態和設定警示、追蹤遊戲伺服器效能和使用量、使用手動或自動調整規模管理容量。

AWS 提供下列監控工具來監看 Amazon GameLift Servers、回報錯誤，並適時採取自動動作：

- Amazon GameLift Servers 主控台 – 使用圖形界面來管理您的 Amazon GameLift Servers 資源並追蹤遊戲託管活動。
- Amazon CloudWatch – 您可以即時監控 Amazon GameLift Servers 指標，以及您在 AWS 服務上執行之其他 AWS 資源和應用程式的指標。CloudWatch 提供一套監控功能，包括建立自訂儀表板的工具，以及在指標達到指定閾值時通知或採取動作的設定警示功能。
- AWS CloudTrail – 會擷取由您的帳戶發出或代表 AWS 您的帳戶發出的所有 API 呼叫 Amazon GameLift Servers 和相關事件和其他 AWS 服務。資料會以日誌檔案形式交付至您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、進行呼叫的來源 IP 地址，以及呼叫的時間。
- 遊戲工作階段日誌 – 您可以輸出遊戲工作階段的自訂伺服器訊息，以記錄存放在 Amazon S3 中的檔案。

監控來源之間的指標比較

Amazon GameLift Servers 透過三個主要來源提供指標：Amazon GameLift Servers 主控台機群活動指標、適用於 Unreal 的 Amazon GameLift Servers 伺服器指標外掛程式，以及 Amazon CloudWatch Amazon GameLift Servers 指標。了解每個來源的重疊和唯一功能，可協助您選擇適合您需求的監控方法。

依來源的指標可用性

下列各節說明不同監控來源之間可用的指標，強調重疊和唯一功能。

所有來源中可用的指標

下列指標可在主控台、外掛程式和 CloudWatch 中一致地使用：

- `Active Instances/ActiveInstances` – 執行作用中伺服器程序且狀態為 `ACTIVE` 的執行個體數量

主控台和 CloudWatch 中可用的指標

下列指標可在 Amazon GameLift Servers 主控台和 Amazon CloudWatch 中使用，提供一致的機群管理和容量監控：

- `Idle instances/IdleInstances` – 目前託管零遊戲工作階段的作用中執行個體
- `Percent idle instances/PercentIdleInstances` – 閒置的作用中執行個體百分比
- `Desired instances/DesiredInstances` – Amazon GameLift Servers 維護的作用中執行個體目標數量
- `Max instances/MaxInstances` – 機群允許的執行個體數目上限
- `Min instances/MinInstances` – 機群允許的執行個體數量下限
- `Instance interruptions/InstanceSpotInterruptions` – 已中斷的 Spot 執行個體數量
- `Recycled instances/RecycledSpotInstances` – 已回收和取代的 Spot 執行個體數量
- `Game session interruptions/GameSessionSpotInterruptions` – 由於 Spot 執行個體中斷而中斷的遊戲工作階段
- `Unhealthy instances replaced/UnhealthyInstancesReplaced` – 由於運作狀態問題而取代的執行個體
- `Activating game sessions` – 目前正在啟動的遊戲工作階段
- `Active game sessions` – 目前正在執行和託管玩家的遊戲工作階段
- `Available game sessions` – 可以接受其他玩家的遊戲工作階段
- `Concurrent activatable game sessions` – 可同時啟動的遊戲工作階段
- `Percent available game sessions` – 新玩家可用的遊戲工作階段百分比

CloudWatch 獨有的指標

Amazon CloudWatch 提供特定的操作指標：

- `Unhealthy instances replaced` – 由於運作狀態問題而取代的執行個體

容器機群獨有的指標

容器機群提供容器化遊戲伺服器特有的其他指標：

容器群組指標

- `ActiveGameServerContainerGroups` – 作用中遊戲伺服器容器群組的數量
- `IdleGameServerContainerGroups` – 閒置遊戲伺服器容器群組的數量
- `PendingGameServerContainerGroups` – 待定遊戲伺服器容器群組的數量
- `TerminatingGameServerContainerGroups` – 終止遊戲伺服器容器群組的數量
- `UnhealthyGameServerContainerGroupsReplaced` – 已取代運作狀態不佳的容器群組數量

容器效能指標

- `ContainerCPUUtilizationPerInstance` – 每個容器執行個體的 CPU 使用率
- `ContainerCPUReservation` – 容器的預留 CPU 容量
- `ContainerMemoryReservation` – 容器的預留記憶體容量
- `ContainerMemoryUtilization` – 容器的記憶體使用率
- `ContainerNetworkIn` – 容器的傳入網路流量
- `ContainerNetworkOut` – 容器的傳出網路流量
- `ContainerStorageReadBytes` – 容器的儲存讀取位元組
- `ContainerStorageWriteBytes` – 容器的儲存寫入位元組

重新命名執行個體指標

- `InstanceCPUUtilization` – 執行個體層級 CPU 使用率（從 `CPUUtilization` 重新命名）
- `InstanceNetworkIn` – 執行個體層級傳入網路流量（從 `NetworkIn` 重新命名）
- `InstanceNetworkOut` – 執行個體層級傳出網路流量（從 `NetworkOut` 重新命名）
- `InstanceStorageReadBytes` – 執行個體層級儲存讀取位元組（從 `DiskReadBytes` 重新命名）
- `InstanceStorageWriteBytes` – 執行個體層級儲存體寫入位元組（從 `DiskWriteBytes` 重新命名）
- `InstanceStorageReadOps` – 執行個體層級儲存體讀取操作（從 `DiskReadOps` 重新命名）
- `InstanceStorageWriteOps` – 執行個體層級儲存體寫入操作（從 `DiskWriteOps` 重新命名）

選擇正確的監控來源

根據您的特定需求選取監控方法：

- 使用 Amazon GameLift Servers 主控台進行機群管理、容量規劃和一般操作監督。主控台提供機群運作狀態和玩家活動的整合檢視。如需詳細資訊，請參閱[the section called “在主控台中追蹤遊戲託管”](#)。
- 使用 Amazon CloudWatch 與其他 AWS 服務進行自動監控、警示和整合。CloudWatch 啟用自訂儀表板和警示型自動化。如需詳細資訊，請參閱[the section called “使用 CloudWatch 監控”](#)。
- 使用多個來源進行全面監控。結合主控台監督、外掛程式技術詳細資訊和 CloudWatch 自動化，以完全了解您的遊戲託管基礎設施。

如需其他監控功能，您也可以使用：

- [the section called “記錄 API 呼叫”](#) – 追蹤 API 呼叫和相關事件以進行稽核和合規。
- [the section called “記錄伺服器訊息”](#) – 擷取自訂伺服器訊息和遊戲工作階段日誌。

主題

- [the section called “在主控台中追蹤遊戲託管”](#)
- [the section called “使用 CloudWatch 監控”](#)
- [the section called “記錄 API 呼叫”](#)
- [the section called “記錄伺服器訊息”](#)

在Amazon GameLift Servers主控台中追蹤遊戲託管

使用Amazon GameLift Servers主控台近乎即時地檢視和管理遊戲託管資源和持續託管活動。主控台為服務 API 的大部分功能提供圖形界面Amazon GameLift Servers。您可以使用 主控台來：

- 將儀表板用於高階快照。您可以查看所有Amazon GameLift Servers託管資源的號碼和目前狀態，並遵循連結以取得個別資源的詳細資訊。
- 管理個別託管資源。您可以建立、檢視和刪除所有Amazon GameLift Servers資源，並更新其可變屬性。您也可以檢視特定類型的託管活動，例如事件和效能指標。

- 與遊戲和玩家工作階段活動互動。您可以依機群追蹤遊戲工作階段和玩家工作階段活動，並使用此資訊來疑難排解遊戲工作階段問題。檢視遊戲工作階段的詳細資訊、檢視每個遊戲工作階段的玩家工作階段，以及在多個遊戲工作階段中查詢玩家活動。您也可以視需要關閉個別遊戲工作階段。

主題

- [在儀表板中託管Amazon GameLift Servers儀表板](#)
- [Amazon GameLift Servers 儀表板中的遊戲伺服器建置](#)
- [Amazon GameLift ServersRealtimeAmazon GameLift Servers主控台中的指令碼](#)
- [Amazon GameLift Servers 主控台中的機群](#)
- [Amazon GameLift Servers 主控台中的機群詳細資訊](#)
- [Amazon GameLift Servers 主控台中的遊戲和玩家工作階段](#)
- [Amazon GameLift Servers 主控台中的別名](#)
- [Amazon GameLift Servers 主控台中的遊戲工作階段佇列](#)

在儀表板中託管Amazon GameLift Servers儀表板

使用 Amazon GameLift Servers主控台儀表板，取得您 AWS 帳戶中Amazon GameLift Servers託管資源目前狀態的高階快照。Amazon GameLift Servers 儀表板提供下列項目的檢視：

- 處於就緒、初始化和失敗狀態的組建數目。選擇檢視組建以取得目前區域中組建的詳細資訊。
- 所有狀態的機群數量。選擇檢視機群以取得目前區域中機群的詳細資訊。
- 您目前的資源。
- 新功能和服務公告。

開啟 Amazon GameLift Servers 儀表板

- 在[Amazon GameLift Servers主控台](#)的導覽窗格中，選擇儀表板。

從儀表板，您可以：

- 選擇準備啟動並填寫對應的啟動問卷，以準備您的遊戲以啟動。
- 選擇檢視服務配額，以請求增加服務配額，以準備啟動或回應啟動。
- 在功能焦點中選擇連結，以檢視部落格文章和新功能的詳細資訊。

Amazon GameLift Servers 儀表板中的遊戲伺服器建置

在[Amazon GameLift Servers主控台](#)的組建頁面上，您可以檢視和管理已上傳到 的所有遊戲伺服器組建的相關資訊，Amazon GameLift Servers以便在受管 EC2 機群上部署。在導覽窗格中，選擇託管、受管 EC2、組建。

組建頁面會顯示每個組建的下列資訊。您可以使用偏好設定工具視需要調整資料表內容（請參閱資料表右上角

的
示）。自訂偏好設定會儲存到您的帳戶 AWS 使用者，並會在您檢視此頁面時自動套用。

圖

Note

建置頁面只會顯示您目前 AWS 區域中的建置。

- 名稱 – 與上傳組建相關聯的名稱。
- 狀態 – 組建的狀態。顯示三個狀態訊息之一：
 - 已初始化 – 上傳尚未開始或仍在進行中。
 - 就緒 – 組建已準備好建立機群。
 - 失敗 – Amazon GameLift Servers在收到二進位檔之前，建置逾時。
- 建立時間 – 您上傳建置至 的日期和時間Amazon GameLift Servers。
- 組建 ID – 在上傳時指派給組建的唯一 ID。
- 版本 – 與上傳組建相關聯的版本標籤。
- 作業系統 – 建置執行所在的作業系統。組建作業系統會決定哪些作業系統Amazon GameLift Servers安裝在機群的執行個體上。
- 大小 – 上傳至 的建置檔案的大小，以 MB (MB) 為單位Amazon GameLift Servers。
- 機群 – 隨組建一起部署的機群數量。

您可以在此頁面進行下列任一操作：

- 查看建置的詳細資訊。選擇組建的名稱以開啟其組建詳細資訊頁面。
- 從建置建立新機群。選取組建，然後選擇建立機群。
- 篩選和排序建置清單。使用表格上方的控制項。

- 刪除建置。選取組建，然後選擇刪除。

組建詳細資訊

在建置頁面上，選擇建置的名稱以開啟其詳細資訊頁面。詳細資訊頁面的概觀區段會顯示與建置頁面相同的建置摘要資訊。機群區段顯示使用組建建立的機群清單，包括與[機群頁面](#)相同的摘要資訊。

Amazon GameLift ServersRealtimeAmazon GameLift Servers主控台中的指令碼

在 [Amazon GameLift Servers主控台](#) 的指令碼頁面上，您可以檢視和管理已上傳至 的所有 Amazon GameLift ServersRealtime 指令碼，Amazon GameLift Servers 以便在受管 EC2 機群上部署。在導覽窗格中，選擇託管、指令碼。

指令碼頁面顯示每個指令碼的下列資訊。您可以使用偏好設定工具視需要調整資料表內容（請參閱資料表右上角

的
示）。自訂偏好設定會儲存到您的帳戶 AWS 使用者，並會在您檢視此頁面時自動套用。

圖

Note

指令碼頁面只會顯示目前 AWS 區域中的指令碼。

- 名稱 – 與上傳的指令碼相關聯的名稱。
- ID – 上傳時指派給指令碼的唯一 ID。
- 版本 – 與上傳指令碼相關聯的版本標籤。
- 大小 – 上傳至 之指令碼檔案的大小，以 MB (MB) 為單位 Amazon GameLift Servers。
- 建立時間 – 您上傳指令碼至 的日期和時間 Amazon GameLift Servers。
- 機群 – 與指令碼一起部署的機群數量。

您可以在此頁面進行下列任一操作：

- 檢視指令碼詳細資訊。選擇組建的名稱以開啟其指令碼詳細資訊頁面。
- 從指令碼建立新機群。選取指令碼，然後選擇建立機群。
- 篩選和排序指令碼清單。使用表格上方的控制項。

- 刪除指令碼。選取指令碼，然後選擇刪除。

指令碼詳細資訊

在指令碼頁面上，選擇指令碼的名稱以開啟其詳細資訊頁面。詳細資訊頁面的概觀區段會顯示與建置頁面相同的指令碼摘要資訊。機群區段顯示使用指令碼建立的機群清單，包括與[機群頁面](#)相同的摘要資訊。

Amazon GameLift Servers 主控台中的機群

您可以在 AWS 您的帳戶 Amazon GameLift Servers 下檢視為託管遊戲而建立的所有機群的相關資訊。此清單顯示已建立您目前區域的機群。從機群頁面，您可以建立新的機群或檢視機群的其他詳細資訊。機群的[詳細資訊頁面](#)包含用量資訊、指標、遊戲工作階段資料和玩家工作階段資料。您也可以編輯機群記錄或刪除機群。

若要檢視機群頁面，請從導覽窗格中選擇機群。

Fleets (機群) 頁面預設會顯示以下摘要資訊。您可以使用偏好設定工具視需要調整資料表內容（請參閱資料表右上角

的
示）。自訂偏好設定會儲存到您的帳戶 AWS 使用者，並會在您檢視此頁面時自動套用。

- ID – 指派給機群的識別符。此 ID 在建立機群的中是唯一 AWS 區域的。
- 名稱 – 提供給機群的易記名稱。
- 狀態 – 機群的状态：新增、下載、建置和作用中。
- 建立時間 – 建立機群的日期和時間。

Note

對於超過 90 天前建立的機群，機群會顯示警告圖示。作為最佳實務，我們建議您每 30 天替換一次機群，以維護託管遊戲伺服器的安全 up-to-date 執行期環境。如需準則，請參閱[Amazon GameLift Servers 的安全最佳實務](#)。

- 機群類型 – 用來託管遊戲的執行個體可用性，這可能會影響託管成本。機群可以使用隨需（一律可用）或 Spot（可用性不同）執行個體。
- 作用中執行個體 – 機群正在使用的 EC2 執行個體數量。
- 所需執行個體 – 保持作用中的 EC2 執行個體數量。

圖

- 遊戲工作階段 – 在機群中執行的作用中遊戲工作階段數目。資料延遲五分鐘。
- 玩家工作階段 – 機群中作用中玩家工作階段的數量。資料延遲五分鐘。

Amazon GameLift Servers 主控台中的機群詳細資訊

透過選擇機群名稱，從儀表板或機群頁面存取機群詳細資訊頁面。

在機群詳細資訊頁面上，您可以採取下列動作：

- 更新機群的屬性、連接埠設定和執行時間組態。
- 新增或移除機群位置。
- 變更機群容量設定。
- 設定或變更目標追蹤自動調整規模。
- 刪除機群。

詳細資訊

機群設定

- 機群 ID – 指派給機群的識別符。此 ID 在建立機群的 中是唯一 AWS 區域 的。
- 機群名稱 – 提供給機群的易記名稱。
- ARN – 指派給此機群的唯一識別符。機群的 ARN 將其識別為 Amazon GameLift Servers 資源，並指定區域和 AWS 帳戶。
- 描述 – 機群的簡短可識別描述。
- 狀態 – 機群的目前狀態，可能是新增、下載、建置和作用中。
- 建立時間 – 建立機群的日期和時間。

Note

對於超過 90 天前建立的機群，機群會顯示警告圖示。作為最佳實務，我們建議您每 30 天替換一次機群，以維護託管遊戲伺服器的安全 up-to-date 執行期環境。如需準則，請參閱 [Amazon GameLift Servers 的安全最佳實務](#)。

- 運算類型 – 用來託管遊戲的運算類型。機群可以是受管 EC2、受管容器或 Anywhere 機群。
- 作業系統 – 部署至機群中每個執行個體的 Amazon Machine Image (AMI)。AMI 版本是建立機群時可用的最新版本。

- 憑證類型 – 機群使用 TLS 憑證來驗證遊戲伺服器 and 加密所有用戶端/伺服器通訊的能力狀態。
- 機群類型 – 用來託管遊戲的執行個體可用性，這可能會影響託管成本。機群可以使用隨需（一律可用）或 Spot（可用性有所不同）執行個體。
- EC2 執行個體類型 – 建立機群時為機群選取的 Amazon EC2 [執行個體類型](#)。
- 執行個體角色 ARN – 如果在機群建立期間提供其他 AWS 資源，則管理其他資源存取權的 AWS IAM 角色。
- 執行個體角色登入資料提供者 – 如果在機群建立期間提供其他 AWS 資源，則管理其他資源存取權的 AWS IAM 角色。
- 指標群組名稱 – 用來彙總多個機群指標的群組。
- 保護政策 – 機群[遊戲工作階段保護](#)的目前設定。
- 每個玩家的遊戲工作階段上限 – 在政策期間，玩家可以建立的工作階段數目上限。
- 政策期間 – 用於限制玩家所建立工作階段數量的時間長度。

組建詳細資訊

組建詳細資訊區段會顯示在機群上託管的組建。選取建置名稱以查看完整的建置詳細資訊頁面。

執行期組態

執行期組態區段會顯示要在每個執行個體上啟動的伺服器程序。它包含遊戲伺服器可執行檔的路徑和選用的啟動參數。

遊戲工作階段啟用

遊戲工作階段啟用區段會顯示同時啟動的伺服器程序數量，以及在終止程序之前等待程序啟用的時間。

EC2 連接埠設定

連接埠區段會顯示機群的連線許可，包括 IP 地址和連接埠設定範圍。

指標

Metrics (指標) 標籤會顯示一段時間內機群指標的圖形表示。如需在 中使用指標的詳細資訊 Amazon GameLift Servers，請參閱 [使用 Amazon CloudWatch 監控 Amazon GameLift Servers](#)。

事件

Events (事件) 標籤提供日誌機群中已發生之所有事件的日誌，包括事件程式碼、訊息和時間戳記。請參閱 Amazon GameLift Servers API 參考中的[事件](#)描述。

擴展

擴展索引標籤包含機群容量的相關資訊，包括目前狀態和容量隨時間變更。它還提供更新容量限制和管理自動調整規模的工具。

擴展容量

檢視每個機群位置的目前機群容量設定。如需變更限制和容量的詳細資訊，請參閱 [使用 擴展遊戲託管容量 Amazon GameLift Servers](#)。

- AWS 位置 – 部署機群執行個體的位置名稱。
- 狀態 – 機群位置的託管狀態。位置狀態必須ACTIVE能夠託管遊戲。
- 最小大小 – 必須部署在該位置的最小執行個體數量。
- 所需執行個體 – 維護位置的作用中執行個體目標數量。當作用中執行個體和所需執行個體不同時，擴展事件會開始視需要啟動或關閉執行個體，直到作用中執行個體等於所需執行個體為止。
- 大小上限 – 可在 位置部署的最多執行個體。
- 可用 – 執行個體的服務限制減去使用中的執行個體數量。此值會告訴您可新增至位置的執行個體數量上限。

自動擴展政策

本節涵蓋套用至機群的自動調整規模政策相關資訊。您可以設定或更新目標型政策。機群的規則型政策必須使用 AWS SDK 或 CLI 定義，如下所示。如需擴展的詳細資訊，請參閱 [使用 Amazon GameLift Servers 自動調整機群容量的規模](#)。

擴展歷史記錄

檢視隨時間變化的容量圖表。

位置

位置索引標籤列出部署機群執行個體的所有位置。位置包括機群的主區域和任何已新增的遠端位置。您可以直接在此索引標籤中新增或移除位置。

- 位置 – 部署機群執行個體的位置名稱。
- 狀態 – 機群位置的託管狀態。位置狀態會追蹤啟用位置中第一個執行個體的程序。位置狀態必須ACTIVE能夠託管遊戲。
- 作用中執行個體 – 機群位置上執行伺服器程序的執行個體數目。
- 作用中伺服器 – 可在機群位置託管遊戲工作階段的遊戲伺服器程序數目。

- 遊戲工作階段 – 機群位置中執行個體上作用中的遊戲工作階段數目。
- 玩家工作階段 – 代表個別玩家的玩家工作階段數量，這些玩家正在參與機群位置中作用中的遊戲工作階段。

遊戲工作階段

Game sessions (遊戲工作階段) 標籤列出過去和目前在機群上託管的遊戲工作階段，包括一些詳細資訊。選擇遊戲工作階段 ID 以存取其他遊戲工作階段資訊，包括玩家工作階段。如需玩家工作階段的詳細資訊，請參閱 [Amazon GameLift Servers 主控台](#) 中的 [遊戲和玩家工作階段](#)。

Amazon GameLift Servers 主控台中的遊戲和玩家工作階段

您可以使用 Amazon GameLift Servers 主控台來使用遊戲工作階段和玩家工作階段。如需遊戲工作階段和玩家工作階段的詳細資訊，請參閱 [玩家如何連接到遊戲](#)。Amazon GameLift Servers 主控台提供的資訊和工具可協助您調查遊戲工作階段的問題。

您可以做什麼：

- 探索在特定機群上託管的遊戲工作階段和玩家工作階段活動。
- 查詢多個機群中特定玩家的遊戲工作階段活動。
- 關閉特定遊戲工作階段。

檢視遊戲工作階段詳細資訊

遊戲工作階段和玩家工作階段資料由託管遊戲工作階段的機群組織。

存取遊戲工作階段和玩家工作階段資訊

1. 在 [Amazon GameLift Servers 主控台](#) 中，開啟左側導覽窗格。選取託管解決方案類型並開啟機群頁面。例如：
 - Hosting、Anywhere、Fleets
 - 託管、受管 EC2、機群
 - 託管、受管容器、機群
2. 每個機群頁面會顯示目前所選機群的清單 AWS 區域。選擇您要檢視遊戲工作階段資料的機群。
3. 在機群的詳細資訊頁面中，開啟遊戲工作階段索引標籤。此索引標籤會列出在機群上託管的所有遊戲工作階段，以及摘要資訊。您可

以使用偏好設定工具視需要調整資料表內容（請參閱資料表右上角的

示）。自訂偏好設定會儲存到您的帳戶 AWS 使用者，並會在您檢視此頁面時自動套用。

4. 從清單中選擇遊戲工作階段，以檢視其他資訊。
5. 如果遊戲工作階段包含玩家工作階段資料，請選擇檢視玩家工作階段，以自動填入遊戲工作階段 ID 來開啟玩家工作階段查詢工具。

遊戲工作階段詳細資訊包含下列資訊：

- 狀態 – 遊戲工作階段狀態。
 - 啟用中 – 執行個體正在啟動遊戲工作階段。
 - 作用中 – 根據工作階段的玩家[建立政策](#)，[遊戲工作階段正在執行並可用於接收玩家](#)。
 - 已終止 – 遊戲工作階段已結束。
- ARN – 遊戲工作階段的 Amazon Resource Name。
- 名稱 – 為遊戲工作階段產生的名稱。
- 位置 – Amazon GameLift Servers託管遊戲工作階段的位置。
- 建立時間 – Amazon GameLift Servers建立串流工作階段的日期和時間。
- 結束時間 – 遊戲工作階段結束的日期和時間。
- DNS 名稱 – 遊戲工作階段的主機名稱。
- IP 地址 – 為遊戲工作階段指定的 IP 地址。
- 連接埠 – 用來連線至遊戲工作階段的連接埠號碼。
- 建立者 ID – 啟動遊戲工作階段之玩家的唯一識別符。
- 玩家工作階段建立政策 – 指出遊戲工作階段是否接受新的玩家。
- 遊戲擴展保護政策 – 在機群中Amazon GameLift Servers啟動的所有新執行個體上設定的遊戲工作階段保護類型。

遊戲資料

遊戲屬性資料，格式為字串，可在開始時傳送至您的遊戲工作階段。

遊戲屬性

遊戲屬性資料，格式為金鑰/值對，可在開始時傳送至您的遊戲工作階段。

配對資料

如果遊戲工作階段是使用 建立FlexMatch，配對資料會說明配對組態和規則集的相關資訊。這包括每個配對的玩家屬性和團隊指派。資料為 JSON 格式。如需FlexMatch配對的詳細資訊，請參閱[建置配對建構器](#)。

查詢玩家工作階段資料

如果您的遊戲託管解決方案使用玩家工作階段並提供唯一的玩家 IDs，您可以探索玩家特定活動，以了解多個機群過去或現在的遊戲工作階段。使用下列其中一種方法開啟玩家工作階段查詢工具：

- 在 Amazon GameLift Servers 主控台中，開啟左側導覽窗格，然後選擇玩家工作階段查詢，然後選取要使用的篩選條件類型。
- 檢視機群的遊戲工作階段詳細資訊時，請選擇檢視玩家工作階段。查詢工具會開啟，遊戲工作階段會預先選取遊戲工作階段 ID 篩選條件，並填入遊戲工作階段值。

使用查詢工具時，您可以提供下列任何資訊：

- 取得特定玩家工作階段相關資訊的玩家工作階段 ID。
- 取得所請求遊戲工作階段之所有玩家工作階段相關資訊的遊戲工作階段 ID。結果代表預留位置或連接到遊戲工作階段的所有玩家。您可以選擇依玩家工作階段狀態篩選結果。
- 玩家 ID，以取得所請求玩家之所有玩家工作階段的資訊。結果代表玩家參與的所有遊戲工作階段。

Note

查詢工具會搜尋目前選取的所有玩家工作階段活動 AWS 區域。如果您在 區域中有多個機群，則結果會包含所有機群的玩家工作階段活動。對於多位置機群，結果也包含機群遠端位置的玩家工作階段活動。

為每個遊戲工作階段收集的玩家工作階段資料如下：

- 玩家工作階段 ID – 指派給玩家工作階段的識別符。
- 玩家 ID – 玩家的唯一識別符。選擇此 ID 以取得其他玩家資訊。
- 遊戲工作階段 ID – 指派給遊戲工作階段的識別符。
- 機群 ID – 指派給託管遊戲工作階段之機群的識別符。
- 狀態 – 玩家工作階段的狀態。以下是可能的狀態：
 - 預留 – 已預留玩家工作階段，但玩家未連線。

- 作用中 – 播放器工作階段已連線至遊戲伺服器。
- 已完成 – 播放器工作階段已結束；播放器不再連線。
- 逾時 – 播放器無法連線。
- 建立時間 – 玩家連線到遊戲工作階段的時間。
- 結束時間 – 玩家從遊戲工作階段中斷連線的時間。
- 連線資料 – 播放器用來連線至遊戲工作階段的 IP 地址、DNS 名稱和連接埠。
- 玩家資料 – 在玩家工作階段建立期間提供的玩家相關資訊。

關閉遊戲工作階段

使用 Amazon GameLift Servers 主控台關閉特定遊戲工作階段。此功能為您提供一種簡單快速的方法，可讓您尋找遊戲工作階段並傳送訊號來終止它。另一種終止方法要求您找到遊戲工作階段正在執行的機群執行個體、遠端存取執行個體，以及手動關閉遊戲工作階段。

您可以基於任何原因關閉遊戲工作階段。最常見的原因是解決無法自然關閉的遊戲工作階段。因此，遊戲工作階段的託管資源無法釋出來說管新的遊戲工作階段，而且機群的託管容量會降低。

Note

此功能依賴於託管解決方案的特定組態設定。它有下列限制：

- 遊戲工作階段必須託管在執行適用於 Amazon GameLift Servers v5 或更高版本之伺服器 SDK 的遊戲伺服器組建的機群上。如果您的遊戲伺服器是以較舊的版本部署，您需要使用遠端存取來刪除遊戲工作階段。
- 如果遊戲工作階段託管在 Anywhere 機群上，機群必須使用 Amazon GameLift Servers 代理程式來管理遊戲伺服器程序。

終止遊戲工作階段

1. 在 [Amazon GameLift Servers 主控台](#) 中，開啟左側導覽窗格。選取託管解決方案類型並開啟機群頁面。例如：
 - 託管、隨處、機群
 - 託管、受管 EC2、機群
 - 託管、受管容器、機群

2. 每個機群頁面會顯示目前所選機群的清單 AWS 區域。選擇託管您要終止之遊戲工作階段的機群。
3. 在機群的詳細資訊頁面中，開啟遊戲工作階段索引標籤。在遊戲工作階段清單中，選取您要終止的工作階段，然後選擇終止按鈕。
4. 在終止遊戲工作階段？視窗中，確認您已關閉正確的遊戲工作階段，並選擇終止方法。
 - 正常遊戲工作階段關閉 – 此選項會將訊號傳送至託管遊戲工作階段以關閉的伺服器程序。如果您的遊戲伺服器建置已針對 正確整合 Amazon GameLift Servers，伺服器程序會啟動其遊戲工作階段關閉順序，Amazon GameLift Servers 並通知其即將結束，然後停止。根據您的遊戲設計，關機順序可能包含正常完成遊戲工作階段的步驟，例如儲存資料和通知作用中玩家。此方法可能需要一小段延遲才能完成遊戲工作階段關閉序列。
 - 立即關閉遊戲工作階段 – 此選項會將訊號傳送至程序管理員，以關閉託管遊戲工作階段的伺服器程序。此選項會略過正常的遊戲工作階段關閉。即使伺服器程序無法回應，也可以終止遊戲工作階段。
5. 確認遊戲工作階段終止。您可以在遊戲工作階段主控台頁面上追蹤關機進度。遊戲工作階段狀態會變更為「終止」，然後在關閉完成時變更為「終止」。

相關主題

- 您也可以使用 AWS SDK 和 關閉遊戲工作階段 AWS CLI。如需詳細資訊和範例，請參閱 Amazon GameLift Servers API 參考主題 [TerminateGameSession](#)。
- 如需遊戲伺服器整合以及伺服器程序如何回應來自 Amazon GameLift Servers 服務之訊號的詳細資訊，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

Amazon GameLift Servers 主控台中的別名

別名頁面會顯示引導流量特定託管目的地的 Amazon GameLift Servers 別名相關資訊。若要別名，請在導覽窗格中選擇託管、別名。

您可以在別名頁面上執行下列動作：

- 建立新的別名。選擇建立別名。
- 篩選和排序別名資料表。使用表格上方的控制項。您可以使用偏好設定工具視需要調整資料表內容（請參閱資料表右上角的圖示）。自訂偏好設定會儲存到您的帳戶 AWS 使用者，並會在您檢視此頁面時自動套用。
- 檢視別名詳細資訊。選擇別名名稱以開啟別名詳細資訊頁面。

圖

- 刪除別名。選擇別名，然後選擇刪除。

別名詳細資訊

別名詳細資訊頁面會顯示別名的相關資訊。

在此頁面上，您可以：

- 編輯別名。選擇編輯。
- 檢視您與別名相關聯的機群。
- 刪除別名。選擇 刪除。

別名詳細資訊包括：

- ID – 用來識別別名的唯一號碼。
- 描述 – 別名的描述。
- ARN – 別名的 Amazon Resource Name。
- 建立 – 別名建立的日期和時間。
- 上次更新 – 上次更新別名的日期和時間。
- 路由類型 – 別名的路由類型，可以是下列其中一項：
 - 簡單 – 將玩家流量路由到指定的機群 ID。您可以隨時更新別名的機群 ID。
 - 終端機 – 將訊息傳遞回用戶端。例如，您可以將使用out-of-date用戶端的玩家導向到可以升級的位置。
- 標籤 – 用於識別別名的金鑰和值對。

Amazon GameLift Servers 主控台中的遊戲工作階段佇列

您可以檢視所有佇列的資訊，這些佇列用於處理新遊戲工作階段的請求。佇列頁面會顯示目前選取之中的遊戲工作階段佇列 AWS 區域。從 Queues (佇列) 頁面，您可以建立新的佇列、刪除現有的佇列，或針對選取的佇列開啟詳細資訊頁面。每個佇列詳細資訊頁面都包含佇列的組態和指標資料。如需佇列的詳細資訊，請參閱[使用Amazon GameLift Servers佇列管理遊戲工作階段置放](#)。

佇列頁面會顯示每個佇列的下列摘要資訊。您可以使用偏好設定工具視需要調整資料表內容（請參閱資料表右上角

的
示）。自訂偏好設定會儲存到您的帳戶 AWS 使用者，並會在您檢視此頁面時自動套用。

圖

- 佇列名稱 – 指派給佇列的名稱。新遊戲工作階段的請求會使用此名稱來指令佇列。
- 佇列逾時 – 遊戲工作階段置放請求在逾時前保留在佇列中的時間長度上限，以秒為單位。
- 佇列中的目的地 – 佇列組態中列出的機群數量。會在佇列中的任何機群上Amazon GameLift Servers放置新的遊戲工作階段。

檢視佇列詳細資訊

您可以存取任何佇列的詳細資訊，包括佇列組態和指標。若要開啟佇列詳細資訊頁面，請前往佇列頁面，然後選擇佇列名稱。

佇列詳細資訊頁面顯示摘要表格和包含額外資訊的標籤。您可以在此頁面進行下列操作：

- 更新佇列的組態、目的地清單和玩家延遲政策。選擇編輯。
- 刪除佇列。刪除佇列後，所有參考該佇列名稱的新遊戲工作階段請求都會失敗。選擇 刪除。

Note

若要還原已刪除的佇列，請使用已刪除佇列的名稱建立新的佇列。

詳細資訊

概觀

概觀區段會顯示佇列的 Amazon Resource Name (ARN) 和逾時。您可以在 的其他動作或區域中參考佇列時使用 ARNAmazon GameLift Servers。逾時是遊戲工作階段置放請求在逾時前保留在佇列中的時間長度上限，以秒為單位。

事件通知

事件通知區段列出 SNS 主題發佈事件通知至 Amazon GameLift Servers，以及新增至此佇列建立之所有事件的事件資料。

標籤

標籤表格會顯示用於標記資源的索引鍵和值。如需標記的詳細資訊，請參閱[標記 AWS 資源](#)。

指標

Metrics (指標) 標籤顯示一段時間內佇列指標的圖形表示。

佇列指標包含描述佇列中置放活動的資訊範圍，包括依區域組織的成功置放。您可以使用區域資料來了解您託管遊戲的位置。區域置放指標有助於偵測整體佇列設計的問題。

Amazon CloudWatch 也提供佇列指標。如需可用指標的說明，請參閱 [佇列的 Amazon GameLift Servers 指標](#)。

目的地

Destinations (目的地) 標籤顯示所有為佇列列出的機群或別名。

當 Amazon GameLift Servers 搜尋目的地是否有可用資源來託管新的遊戲工作階段時，它會搜尋此處列出的預設順序。只要列出的第一個目的地有容量，就會將新的遊戲工作階段 Amazon GameLift Servers 放在該處。透過提供玩家延遲資料，您可以讓個別遊戲工作階段放置請求覆寫預設的順序。此資料告知 Amazon GameLift Servers 搜尋可用的最低平均玩家延遲目的地。如需設計佇列的詳細資訊，請參閱 [自訂遊戲工作階段佇列](#)。

工作階段置放

玩家延遲政策

播放器延遲政策區段會顯示佇列使用的所有政策。資料表會依政策強制執行的順序列出政策。

位置

位置區段顯示此佇列可以放置遊戲工作階段的位置。

優先順序

優先順序區段顯示佇列評估遊戲工作階段詳細資訊的順序。

位置順序

位置順序區段顯示佇列在放置遊戲工作階段時使用的預設順序。如果您尚未定義其他類型的優先順序，佇列會使用此順序。

使用 Amazon CloudWatch 監控 Amazon GameLift Servers

您可以使用 Amazon GameLift Servers Amazon CloudWatch 進行監控，該 AWS 服務會收集原始資料並將其處理為可讀且近乎即時的指標。這些統計資料會保留 15 個月，以提供遊戲伺服器託管與 Amazon GameLift Servers 搭配之執行方式的歷史觀點。您可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需更多資訊，請參閱 [Amazon CloudWatch 使用者指南](#)。

下表列出 Amazon GameLift Servers 的指標和維度。CloudWatch 中提供的所有指標也可在 Amazon GameLift Servers 主控台中使用，該主控台提供一組可自訂圖形的資料。若要存取遊戲的 CloudWatch 指標，請使用 AWS Management Console、AWS CLI、或 CloudWatch API。

如果指標沒有位置，則會使用主位置。

Amazon GameLift Servers 指標的維度

Amazon GameLift Servers 支援以下列維度篩選指標。

維度	描述
Location	篩選機群部署位置的指標。如果指標沒有位置，則會使用主位置。
FleetId	篩選單一機群的指標。此維度可用於執行個體、伺服器程序、遊戲工作階段及玩家工作階段的所有機群指標。
MetricGroup	篩選機群集合的指標。藉由將指標群組名稱新增至機群的屬性，即可將該機群新增至指標群組中 (請參閱 UpdateFleetAttributes())。此維度可用於執行個體、伺服器程序、遊戲工作階段及玩家工作階段的所有機群指標。
QueueName	篩選單一佇列的指標。此維度僅用於遊戲工作階段佇列的指標。
ConfigurationName	篩選單一配對組態的指標。此維度僅用於配對組態的指標。
ConfigurationName-RuleName	篩選配對組態與配對規則之交集的指標。此維度僅用於配對的指標。
InstanceType	篩選 EC2 執行個體類型名稱 (例如「c4.large」) 的指標。此維度用於 Spot 執行個體的指標。
OperatingSystem	篩選執行個體作業系統的指標。此維度與 Spot 執行個體的指標搭配使用。

維度	描述
GameServerGroup	篩選 FleetIQ 遊戲伺服器群組的指標。
ContainerGroupType	篩選容器機群指標，以區分機群位置上的 GameServer ContainerGroup 指標和機群位置上的 PerInstance ContainerGroup 指標。某些容器機群指標 (例如 ContainerNetworkIn 、 ContainerStorageReadBytes) 包含此維度，可讓您區分不同容器群組類型的指標。

機群的 Amazon GameLift Servers 指標

AWS/GameLift 命名空間包含下列與單一機群或機群群組有關的活動的指標：機群與受管理的 Amazon GameLift Servers 解決方案搭配使用。Amazon GameLift Servers 服務每分鐘將指標傳送至 CloudWatch。

執行個體

指標	描述
ActiveInstances	具有 ACTIVE 狀態的執行個體，表示它們正在執行作用中的伺服器程序。此計數包含閒置的執行個體以及託管一或多個遊戲工作階段的執行個體。此指標測量目前的執行個體總容量。此指標可搭配使用自動調整規模。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
DesiredInstances	Amazon GameLift Servers 正在機群中進行維護的作用中執行個體的目標數量。使用自動調整規模時，此值依據目前實施中的調整政策來決定。若無自動調

指標	描述
	整規模，此值以手動設定。檢視機群指標群組的資料時，無法使用此指標。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
IdleInstances	目前託管零 (0) 個遊戲工作階段的作用中執行個體。此指標測量可用但未使用的容量。此指標可搭配使用自動調整規模。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
MaxInstances	設定機群允許的執行個體最大數量。機群的執行個體最大數量可決定手動或自動調整擴展規模時的容量上限。檢視機群指標群組的資料時，無法使用此指標。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置

指標	描述
MinInstances	機群允許的執行個體最小數量。機群的執行個體最小數量可決定手動或自動調整縮減規模時的容量下限。檢視機群指標群組的資料時，無法使用此指標。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
PercentIdleInstances	所有閒置的作用中執行個體的百分比 (計算方式為 $\text{IdleInstances} / \text{ActiveInstances}$)。此指標可用於自動調整規模。 單位：百分比 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
RecycledSpotInstances	已回收和取代的 Spot 執行個體數量。會 Amazon GameLift Servers 回收目前未託管遊戲工作階段，且中斷機率很高的 Spot 執行個體。 單位：計數 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：位置

指標	描述
InstanceSpotInterruptions	已中斷的 Spot 執行個體的數量。 單位：計數 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：位置
CPUUtilization	EC2 指標。此 Amazon GameLift Servers 指標代表機群位置中所有作用中執行個體的硬體效能。Amazon EC2 用來執行執行個體的實體 CPU 時間百分比，其中包括同時執行使用者程式碼和 Amazon EC2 程式碼所花費的時間。由於舊式裝置模擬、非舊式裝置組態、頻繁中斷的工作負載、即時遷移和即時更新等因素，作業系統中工具顯示的百分比可能與 CloudWatch 不同。 單位：百分比
NetworkIn	EC2 指標。此 Amazon GameLift Servers 指標代表機群位置中所有作用中執行個體的硬體效能。執行個體在所有網路界面上收到的位元組數目。此指標可識別流向單一執行個體應用程式的傳入網路流量磁碟區。 單位：位元組
NetworkOut	EC2 指標。此 Amazon GameLift Servers 指標代表機群位置中所有作用中執行個體的硬體效能。執行個體在所有網路界面上送出的位元組數目。此指標可識別流向單一執行個體應用程式的傳出網路流量磁碟區。 單位：位元組

指標	描述
DiskReadBytes	EC2 指標。此Amazon GameLift Servers指標代表機群位置中所有作用中執行個體的硬體效能。從執行個體可用之所有執行個體存放區磁碟區讀取的位元組。此指標用來判斷應用程式從執行個體硬碟中讀取的資料磁碟區。您可以使用它來判斷應用程式的速度。 單位：位元組
DiskWriteBytes	EC2 指標。此Amazon GameLift Servers指標代表機群位置中所有作用中執行個體的硬體效能。寫入至執行個體可用之所有執行個體存放區磁碟區的位元組。此指標用來判斷應用程式寫入至執行個體硬碟中的資料磁碟區。您可以使用它來判斷應用程式的速度。 單位：位元組
DiskReadOps	EC2 指標。此Amazon GameLift Servers指標代表機群位置中所有作用中執行個體的硬體效能。在指定期間，執行個體可用之所有執行個體存放區磁碟區的已完成讀取操作。若要計算該期間的每秒平均 I/O 操作數 (IOPS)，請將該期間的總操作數除以該期間的秒數。 單位：計數
DiskWriteOps	EC2 指標。此Amazon GameLift Servers指標代表機群位置中所有作用中執行個體的硬體效能。在指定期間，執行個體可用之所有執行個體存放區磁碟區的已完成寫入操作。若要計算該期間的每秒平均 I/O 操作數 (IOPS)，請將該期間的總操作數除以該期間的秒數。 單位：計數

伺服器程序

指標	描述
ActiveServerProcesses	具有 ACTIVE 狀態的伺服器程序，表示它們正在執行並可託管遊戲工作階段。此計數包含閒置的伺服器程序以及託管遊戲工作階段的伺服器程序。此指標測量目前的伺服器程序總容量。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
HealthyServerProcesses	報告為健康的作用中伺服器程序。此指標對於追蹤機群的遊戲伺服器整體健康狀況很有用。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
PercentHealthyServerProcesses	所有報告為健康的作用中伺服器程序的百分比 (計算方式為 $\text{HealthyServerProcesses} / \text{ActiveServerProcesses}$)。 單位：百分比 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
ServerProcessAbnormalTerminations	從上次報告之後，因異常情況而被關閉的伺服器程序。此指標包含 Amazon GameLift Servers 服務所起始的終止。當伺服器程序停止回應，並且始終

指標	描述
	報告運作狀態檢查失敗或未徹底終止時 (藉由呼叫 ProcessEnding())，就會發生上述情況。 單位：計數 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：位置
ServerProcessActivations	從上次報告之後，從 ACTIVATING 成功轉換為 ACTIVE 狀態的伺服器程序。伺服器程序必須是作用中，才能託管遊戲工作階段。 單位：計數 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：位置
ServerProcessTerminations	從上次報告之後，已關閉的伺服器程序。這包括因任何原因轉換為 TERMINATED 狀態的所有伺服器程序，包括正常與異常程序終止。 單位：計數 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：位置

遊戲工作階段

指標	描述
ActivatingGameSessions	具有 ACTIVATING 狀態的遊戲工作階段，表示它們正在進行啟動程序。遊戲工作階段必須是作用中，才能

指標	描述
	託管玩家。非常長的持續時間可能表示遊戲工作階段並未從 ACTIVATING 轉換為 ACTIVE 狀態。此指標可搭配使用自動調整規模。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
ActiveGameSessions	具有 ACTIVE 狀態的遊戲工作階段，表示它們能夠託管玩家，並且正在託管零個或更多的玩家。此指標測量目前被託管的遊戲工作階段的總數。此指標可搭配使用自動調整規模。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置

指標	描述
AvailableGameSessions	目前尚未用於託管遊戲工作階段且運作狀態良好的作用中伺服器程序，可以啟動新的遊戲工作階段，而不會延遲啟動新的伺服器程序或執行個體。此指標可搭配使用自動調整規模。  Note 對於限制並行遊戲工作階段啟用的機群，請使用 指標 <code>ConcurrentActivatableGameSessions</code> 。該指標更準確地代表可在不延遲任何類型的情況下啟動的新遊戲工作階段數量。
	單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置

指標	描述
ConcurrentActivatableGameSessions	作用中且運作狀態良好的伺服器程序，目前尚未用於託管遊戲工作階段，可以立即啟動新的遊戲工作階段。 此指標AvailableGameSessions 與下列方式不同：由於遊戲工作階段啟用的限制，它不會計算目前無法啟用新遊戲工作階段的伺服器程序。(請參閱機群 RuntimeConfiguration 選用設定 MaxConcurrentGameSessionActivations)。對於不限制遊戲工作階段啟用的機群，此指標與 相同AvailableGameSessions 。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：位置
PercentAvailableGameSessions	目前未使用的所有作用中伺服器程序 (正常或狀況不良) 上的遊戲工作階段插槽的百分比 (計算方式為 AvailableGameSessions / [ActiveGameSessions + AvailableGameSessions + unhealthy server processes])。此指標可搭配使用自動調整規模。 單位：百分比 相關 CloudWatch 統計資訊：Average 維度：位置

指標	描述
GameSessionInterruptions	已中斷的 Spot 執行個體上的遊戲工作階段數量。 單位：計數 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：位置

玩家工作階段

指標	描述
CurrentPlayerSessions	具有 ACTIVE 狀態 (玩家已連線至作用中遊戲工作階段) 或 RESERVED 狀態 (玩家已在遊戲工作階段中取得插槽，但尚未連線) 的玩家工作階段。此指標可搭配使用自動調整規模。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum
PlayerSessionActivations	從上次報告之後，從 RESERVED 轉換為 ACTIVE 狀態的玩家工作階段。這發生在玩家成功連線至作用中遊戲工作階段時。 單位：計數 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum

Container Fleet 指標

Container Fleet 指標提供容器型遊戲託管的監控資料，包括容器群組狀態、資源使用率和效能指標。

指標	描述
ActiveGameServerContainerGroups	目前作用中且準備好託管遊戲工作階段的容器群組。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：FleetID 和位置
IdleGameServerContainerGroups	作用中但目前未託管遊戲工作階段的容器群組。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：FleetID 和位置
PendingGameServerContainerGroups	正在啟動的容器群組。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：FleetID 和位置
TerminatingGameServerContainerGroups	正在關閉的容器群組。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：FleetID 和位置
UnhealthyGameServerContainerGroupsReplaced	由於運作狀態檢查失敗而取代的容器群組。 單位：計數

指標	描述
	相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：FleetID 和位置
ContainerCPUUtilizationPerInstance	執行個體上所有容器的平均 CPU 使用率。 單位：百分比 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：FleetID 和位置
ContainerCPUReservation	容器預留的 CPU 資源百分比。 單位：百分比 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：FleetID 和位置
ContainerMemoryReservation	容器預留的記憶體資源百分比。 單位：百分比 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：FleetID 和位置
ContainerMemoryUtilization	容器正在使用的預留記憶體百分比。 單位：百分比 相關 CloudWatch 統計資訊：Average、Minimum、Maximum 維度：FleetID 和位置

指標	描述
ContainerNetworkIn	容器接收的網路位元組。 單位：位元組 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：FleetId、位置和 ContainerGroupType
ContainerNetworkOut	容器傳送的網路位元組。 單位：位元組 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：FleetId、位置和 ContainerGroupType
ContainerStorageReadBytes	容器從儲存體讀取的位元組。 單位：位元組 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：FleetId、位置和 ContainerGroupType
ContainerStorageWriteBytes	容器寫入儲存體的位元組。 單位：位元組 相關 CloudWatch 統計資訊：Sum、Average、Minimum、Maximum 維度：FleetId、位置和 ContainerGroupType

佇列的 Amazon GameLift Servers 指標

Amazon GameLift 命名空間包含下列與遊戲工作階段配置佇列有關的活動的指標：佇列與受管理的 Amazon GameLift Servers 解決方案搭配使用。Amazon GameLift Servers 服務每分鐘將指標傳送至 CloudWatch。

指標	描述
AverageWaitTime	在狀態為 PENDING 的佇列中，遊戲工作階段配置請求等待完成的平均時間。 單位：秒 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum 維度：位置
FirstChoiceNotViable	已成功放置但未放置在第一選擇機群的遊戲工作階段，因為該機群未被視為可行 (例如具有高中斷率的 Spot 機群)。此指標是根據成本，而非延遲。第一個選擇的機群是佇列中列出的第一個機群，或者當置放請求包含玩家延遲資料時，它是 FleetIQ 優先順序 選擇的第一個機群。如果沒有可用的 spot 機群，則可以選擇在該區域中的任何機群。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum
FirstChoiceOutOfCapacity	已成功放置但未放置在第一選擇機群的遊戲工作階段，因為該機群沒有可用資源。第一個選擇的機群是佇列中列出的第一個機群，或者當置放請求包含玩家延遲資料時，它是您定義的 FleetIQ 優先順序選擇的第一個機群。 單位：計數

指標	描述
	相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum
LowestLatencyPlacement	已成功放置在為玩家提供佇列之最低可能延遲區域中的遊戲工作階段。只有在放置請求中包括玩家延遲資料時才會發出此指標。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum
LowestPricePlacement	已成功放置在機群中的遊戲工作階段，其佇列為所選區域的最低可能價格。如果佇列沒有 Spot 執行個體，此機群可以是 Spot 機群或隨需執行個體。只有在放置請求中包括玩家延遲資料時才會發出此指標。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum
Placement <region name>	已成功放置於位於指定區域中機群的遊戲工作階段。此指標會依區域劃分 PlacementsSucceeded 指標。 單位：計數 相關 CloudWatch 統計資訊：Sum
PlacementsCanceled	從上次報告之後，在逾時之前被取消的遊戲工作階段配置請求。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum

指標	描述
PlacementsFailed	從上次報告之後，因任何原因失敗的遊戲工作階段配置請求。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum
PlacementsStarted	從上次報告之後，被新增至佇列的新遊戲工作階段配置請求。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum
PlacementsSucceeded	從上次報告之後，導致新遊戲工作階段的遊戲工作階段配置請求。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum
PlacementsTimedOut	從上次報告之後，達到佇列的逾時限制且未完成的遊戲工作階段配置請求。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum

指標	描述
QueueDepth	遊戲工作階段配置請求在狀態為 PENDING 的佇列中的數量。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum 維度：位置

配對的 Amazon GameLift Servers 指標

Amazon GameLift Servers 命名空間包含配對組態和配對規則的 FlexMatch 活動指標。FlexMatch 配對會與受管理的 Amazon GameLift Servers 解決方案搭配使用。Amazon GameLift Servers 服務每分鐘將指標傳送至 CloudWatch。

如需配對活動順序的詳細資訊，請參閱 [Amazon GameLift Servers FlexMatch 運作方式](#)。

配對組態

指標	描述
CurrentTickets	目前正在處理或等待處理的配對請求。 單位：計數 相關 CloudWatch 統計資訊：Average、Minimum、Maximum、Sum
MatchAcceptancesTimedOut	針對需要接受的配對組態，從上次報告之後，在接受過程中逾時的潛在配對。 單位：計數 相關 CloudWatch 統計資訊：Sum
MatchesAccepted	針對需要接受的配對組態，從上次報告之後，已被接受的潛在配對。

指標	描述
	單位：計數 相關 CloudWatch 統計資訊：Sum
MatchesCreated	從上次報告之後，已建立的潛在匹配。 單位：計數 相關 CloudWatch 統計資訊：Sum
MatchesPlaced	從上次報告之後，已成功配置於遊戲工作階段的配對。 單位：計數 相關 CloudWatch 統計資訊：Sum
MatchesRejected	針對需要接受的配對組態，從上次報告之後，至少已被一位玩家拒絕的潛在配對。 單位：計數 相關 CloudWatch 統計資訊：Sum
PlayersStarted	從上次報告之後，已新增的配對票券中的玩家。 單位：計數 相關 CloudWatch 統計資訊：Sum
TicketsFailed	從上次報告之後，導致失敗的配對請求。 單位：計數 相關 CloudWatch 統計資訊：Sum
TicketsStarted	從上次報告之後，已建立的新配對請求。 單位：計數 相關 CloudWatch 統計資訊：Sum

指標	描述
TicketsTimedOut	從上次報告之後，已達到逾時限制的配對請求。 單位：計數 相關 CloudWatch 統計資訊：Sum
TimeToMatch	針對在上次報告之前，已放入潛在配對的配對請求，其票券建立到潛在配對建立之間的時間。 單位：秒 相關 CloudWatch 統計資料：資料範例、平均值、最小值、最大值
TimeToTicketCancel	針對在上次報告之前已取消的配對請求，其票券建立到取消之間的時間。 單位：秒 相關 CloudWatch 統計資料：資料範例、平均值、最小值、最大值
TimeToTicketSuccess	針對在上次報告之前已成功的配對請求，其票券建立到成功配對配置之間的時間。 單位：秒 相關 CloudWatch 統計資料：資料範例、平均值、最小值、最大值

配對規則

指標	描述
RuleEvaluationsPassed	從上次報告之後，已傳送的配對程序過程中的規則評估。此指標僅限於前 50 個規則。 單位：計數

指標	描述
	相關 CloudWatch 統計資訊：Sum
RuleEvaluationsFailed	從上次報告之後，失敗的配對程序過程中的規則評估。此指標僅限於前 50 個規則。 單位：計數 相關 CloudWatch 統計資訊：Sum

FleetIQ 的 Amazon GameLift Servers 指標

Amazon GameLift Servers 命名空間包含 FleetIQ 遊戲伺服器群組和遊戲伺服器活動的指標，作為遊戲託管 FleetIQ 獨立解決方案的一部分。Amazon GameLift Servers 服務每分鐘將指標傳送至 CloudWatch。另請參閱《Amazon EC2 [Auto Scaling 使用者指南](#)》中的使用 [amazon CloudWatch 監控 Auto Scaling 群組和執行個體](#)。Amazon EC2 Auto Scaling

指標	描述
AvailableGameServers	可用來進行遊戲執行的遊戲伺服器，而且目前未被遊戲佔用的遊戲伺服器。此數字包括已佔用但仍處於「可用」狀態的遊戲伺服器。 單位：計數 相關的 Amazon CloudWatch 統計資料：總和 維度：GameServerGroup
UtilizedGameServers	目前遊戲已佔用的遊戲伺服器。這個數字包括處於 UTILIZED 狀態的遊戲伺服器。 單位：計數 相關的 Amazon CloudWatch 統計資料：總和 維度：GameServerGroup

指標	描述
DrainingAvailableGameServers	預計終止之執行個體上的遊戲伺服器目前不支援遊戲。這些遊戲伺服器是因應新佔用要求而佔用的最低優先順序。 單位：計數 相關的 Amazon CloudWatch 統計資料：總和 維度：GameServerGroup
DrainingUtilizedGameServers	預計終止之執行個體上的遊戲伺服器目前支援遊戲。 單位：計數 相關的 Amazon CloudWatch 統計資料：總和 維度：GameServerGroup
PercentUtilizedGameServers	目前支援遊戲執行的遊戲伺服器部分。這項指標表示目前使用中的遊戲伺服器容量。這對於驅動可動態新增和刪除執行個體以符合玩家需求的 Auto Scaling 政策很有用。 單位：百分比 相關的 Amazon CloudWatch 統計資料：平均值、最小值、最大值 維度：GameServerGroup
GameServerInterruptions	Spot 執行個體上的遊戲伺服器由於 Spot 可用性有限而中斷。 單位：計數 相關的 Amazon CloudWatch 統計資料：總和 維度：GameServerGroup、InstanceType

指標	描述
InstanceInterruptions	Spot 執行個體由於可用性有限而中斷。
	單位：計數
	相關的 Amazon CloudWatch 統計資料：總和
	維度：GameServerGroup、InstanceType

使用 記錄 Amazon GameLift Servers API 呼叫 AWS CloudTrail

Amazon GameLift Servers 已與 服務整合 AWS CloudTrail，此服務提供由使用者、角色或 AWS 服務在 中採取之動作的記錄 Amazon GameLift Servers。CloudTrail 會將 Amazon GameLift Servers 的所有 API 呼叫擷取為事件。擷取的呼叫包括從 Amazon GameLift Servers 主控台進行的呼叫，以及針對 Amazon GameLift Servers API 操作的程式碼呼叫。如果您建立追蹤，就可以將 CloudTrail 事件持續交付到 Amazon S3 儲存貯體，包括 Amazon GameLift Servers 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台內的 Event history (事件歷史記錄) 檢視最新事件。您可以利用 CloudTrail 所收集的資訊來判斷向 Amazon GameLift Servers 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

若要進一步了解 CloudTrail，請參閱[AWS CloudTrail 《使用者指南》](#)。

CloudTrail 中的 Amazon GameLift Servers 資訊

當您建立帳戶 AWS 帳戶 時，您的 上會啟用 CloudTrail。當活動在 中發生時 Amazon GameLift Servers，該活動會記錄在 CloudTrail 事件中，以及事件歷史記錄中的其他服務 AWS 事件。您可以在 中檢視、搜尋和下載最近的事件 AWS 帳戶。如需詳細資訊，請參閱「[使用 CloudTrail 事件歷史記錄檢視事件](#)」。

如需您 AWS 帳戶帳戶中正在進行事件的記錄 (包含 Amazon GameLift Servers 的事件)，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。依預設，當您在主控台中建立追蹤時，該追蹤會套用至所有的 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- [建立追蹤的概觀](#)
- [CloudTrail 支援的服務和整合](#)

- [設定 CloudTrail 的 Amazon SNS 通知](#)
- [接收多個區域的 CloudTrail 日誌檔案](#)和[接收多個帳戶的 CloudTrail 日誌檔案](#)

CloudTrail 會記錄所有 Amazon GameLift Servers 動作，列在 [Amazon GameLift Servers API 參考](#)中。例如，對的呼叫CreateGameSessionCreatePlayerSession和UpdateGameSession動作會在 CloudTrail 日誌檔案中產生項目。

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：

- 請求是否使用根或 AWS Identity and Access Management (IAM) 使用者登入資料提出。
- 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。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱 [CloudTrail userIdentity 元素](#)。

了解 Amazon GameLift Servers 日誌檔案項目

追蹤是一種組態，可讓事件以日誌檔案的形式交付至您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的是展示 CreateFleet 和 DescribeFleetAttributes 動作的 CloudTrail 記錄項目。

```
{
  "Records": [
    {
      "eventVersion": "1.04",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/myUserName",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "myUserName"
      },
      "eventTime": "2015-12-29T23:40:15Z",
      "eventSource": "gamelift.amazonaws.com",
      "eventName": "CreateFleet",
```

```

    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "[]",
    "requestParameters": {
      "buildId": "build-92b6e8af-37a2-4c10-93bd-4698ea23de8d",
      "eC2InboundPermissions": [
        {
          "ipRange": "10.24.34.0/23",
          "fromPort": 1935,
          "protocol": "TCP",
          "toPort": 1935
        }
      ],
      "logPaths": [
        "C:\\game\\serverErr.log",
        "C:\\game\\serverOut.log"
      ],
      "eC2InstanceType": "c5.large",
      "serverLaunchPath": "C:\\game\\MyServer.exe",
      "description": "Test fleet",
      "serverLaunchParameters": "-paramX=baz",
      "name": "My_Test_Server_Fleet"
    },
    "responseElements": {
      "fleetAttributes": {
        "fleetId": "fleet-0bb84136-4f69-4bb2-bfec-a9b9a7c3d52e",
        "serverLaunchPath": "C:\\game\\MyServer.exe",
        "status": "NEW",
        "logPaths": [
          "C:\\game\\serverErr.log",
          "C:\\game\\serverOut.log"
        ],
        "description": "Test fleet",
        "serverLaunchParameters": "-paramX=baz",
        "creationTime": "Dec 29, 2015 11:40:14 PM",
        "name": "My_Test_Server_Fleet",
        "buildId": "build-92b6e8af-37a2-4c10-93bd-4698ea23de8d"
      }
    },
    "requestID": "824a2a4b-ae85-11e5-a8d6-61d5cafb25f2",
    "eventID": "c8fbea01-fbf9-4c4e-a0fe-ad7dc205ce11",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  },

```

```
{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/myUserName",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "myUserName"
  },
  "eventTime": "2015-12-29T23:40:15Z",
  "eventSource": "gamelift.amazonaws.com",
  "eventName": "DescribeFleetAttributes",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "[]",
  "requestParameters": {
    "fleetIds": [
      "fleet-0bb84136-4f69-4bb2-bfec-a9b9a7c3d52e"
    ]
  },
  "responseElements": null,
  "requestID": "82e7f0ec-ae85-11e5-a8d6-61d5cafb25f2",
  "eventID": "11daabcb-0094-49f2-8b3d-3a63c8bad86f",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
},
]
```

在中記錄伺服器訊息 Amazon GameLift Servers

您可以從日誌檔中的Amazon GameLift Servers伺服器擷取自訂伺服器訊息。您設定記錄的方式取決於您使用的是自訂伺服器或 Amazon GameLift ServersRealtime (請參閱本章中的適當小節)。

主題

- [記錄伺服器訊息 \(自訂伺服器 \)](#)
- [記錄伺服器訊息 \(Amazon GameLift ServersRealtime\)](#)

記錄伺服器訊息（自訂伺服器）

您可以從日誌檔案中的自訂伺服器擷取Amazon GameLift Servers自訂伺服器訊息。若要了解 的記錄 Amazon GameLift ServersRealtime，請參閱 [記錄伺服器訊息 \(Amazon GameLift ServersRealtime\)](#)。

Important

每個遊戲工作階段の日誌檔案大小都有限制（請參閱 中的[Amazon GameLift Servers端點和配額AWS](#) 一般參考）。當遊戲工作階段結束時，會將伺服器日誌Amazon GameLift Servers上傳至 Amazon Simple Storage Service (Amazon S3)。Amazon GameLift Servers不會上傳超過限制的日誌。日誌可以快速成長並超過大小限制。您應該監控日誌，並將日誌輸出限制為僅必要訊息。

設定自訂伺服器的記錄

使用Amazon GameLift Servers自訂伺服器時，您可以編寫自己的程式碼來執行記錄，這是您在伺服器程序組態中設定的一部分。Amazon GameLift Servers會使用記錄組態來識別它必須在每個遊戲工作階段結束時上傳至 Amazon S3 的檔案。

下列指示說明如何使用簡化的程式碼範例來設定記錄：

C++

設定記錄 (C++)

1. 建立字串向量，其為遊戲伺服器日誌檔案的目錄路徑。

```
std::string serverLog("serverOut.log");           // Example server log file
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
```

2. 提供向量做為 ProcessParameters 物件的 [LogParameters](#)。 [ProcessParameters](#)

```
Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
        listenPort,
```

```
Aws::GameLift::Server::LogParameters(logPaths));
```

3. 當您呼叫 `ProcessReady()` 時，請提供 [ProcessParameters](#) 物件。 [ProcessReady](#)

```
Aws::GameLift::GenericOutcome outcome =  
    Aws::GameLift::Server::ProcessReady(processReadyParameter);
```

如需更完整的範例，請參閱 [ProcessReady\(\)](#)。

C#

設定記錄 (C#)

1. 建立做為遊戲伺服器日誌檔案目錄路徑的字串清單。

```
List<string> logPaths = new List<string>();  
logPaths.Add("C:\\\\game\\\\serverOut.txt");    // Example of a log file that the  
game server writes
```

2. 提供您的清單做為 `ProcessParameters` 物件的 [LogParameters](#)。 [ProcessParameters](#)

```
var processReadyParameter = new ProcessParameters(  
    this.OnGameSession,  
    this.OnProcessTerminate,  
    this.OnHealthCheck,  
    this.OnGameSessionUpdate,  
    port,  
    new LogParameters(logPaths));
```

3. 當您呼叫 `ProcessReady()` 時，請提供 [ProcessParameters](#) 物件。 [ProcessReady](#)

```
var processReadyOutcome =  
    GameLiftServerAPI.ProcessReady(processReadyParameter);
```

如需更完整的範例，請參閱 [ProcessReady\(\)](#)。

寫入日誌

您的日誌檔案會在伺服器程序啟動後存在。您可以使用任何方法來寫入檔案，以寫入日誌。若要擷取伺服器的所有標準輸出和錯誤輸出，請將輸出串流重新映射至日誌檔案，如下列範例所示：

C++

```
std::freopen("serverOut.log", "w+", stdout);  
std::freopen("serverErr.log", "w+", stderr);
```

C#

```
Console.SetOut(new StreamWriter("serverOut.txt"));  
Console.SetError(new StreamWriter("serverErr.txt"));
```

存取伺服器日誌

當遊戲工作階段結束時，Amazon GameLift Servers會自動將日誌存放在 Amazon S3 儲存貯體中，並保留 14 天。若要取得遊戲工作階段日誌的位置，您可以使用 [GetGameSessionLogUrl](#) API 操作。若要下載日誌，請使用 操作傳回的 URL。

記錄伺服器訊息 (Amazon GameLift ServersRealtime)

您可以從日誌檔案中Amazon GameLift ServersRealtime的 擷取自訂伺服器訊息。若要了解自訂伺服器的記錄，請參閱 [記錄伺服器訊息 \(自訂伺服器\)](#)。

有不同類型的訊息可以投射到日誌檔案（請參閱 [在伺服器指令碼中記錄訊息](#)）。除了您的自訂訊息之外，您的Amazon GameLift ServersRealtime輸出系統訊息會使用相同的訊息類型，並寫入相同的日誌檔案。您可以調整機群的記錄層級，以減少伺服器產生的記錄訊息數量（請參閱 [調整記錄層級](#)）。

Important

每個遊戲工作階段的日誌檔案大小都有限制（請參閱 中的[Amazon GameLift Servers端點和配額AWS](#) 一般參考）。當遊戲工作階段結束時，會將伺服器日誌Amazon GameLift Servers上傳至 Amazon Simple Storage Service (Amazon S3)。Amazon GameLift Servers不會上傳超過限制的日誌。日誌可以快速成長並超過大小限制。您應該監控日誌，並將日誌輸出限制為僅必要訊息。

在伺服器指令碼中記錄訊息

您可以在 [指令碼中輸出自訂訊息Amazon GameLift ServersRealtime](#)。使用下列步驟將伺服器訊息傳送至日誌檔案：

1. 建立 `variable` 以保留記錄器物件的參考。

```
var logger;
```

2. 在 `init()` 函數中，從工作階段物件取得記錄器，並將其指派給您的記錄器變數。

```
function init(rtSession) {  
    session = rtSession;  
    logger = session.getLogger();  
}
```

3. 在記錄器上呼叫適當的 函數以輸出訊息。

偵錯訊息

```
logger.debug("This is my debug message...");
```

資訊性訊息

```
logger.info("This is my info message...");
```

警告訊息

```
logger.warn("This is my warn message...");
```

錯誤訊息

```
logger.error("This is my error message...");
```

嚴重錯誤訊息

```
logger.fatal("This is my fatal error message...");
```

客戶遇到嚴重錯誤訊息

```
logger.cxfatal("This is my customer experience fatal error message...");
```

如需指令碼中記錄陳述式的範例，請參閱 [Amazon GameLift Servers Realtime 指令碼範例](#)。

日誌檔案中的輸出指出訊息類型 (DEBUG、INFO、WARN、ERRORFATAL、CXFATAL)，如範例日誌的下列幾行所示：

```
09 Sep 2021 11:46:32,970 [INFO] (gamelift.js) 215: Calling GameLiftServerAPI.InitSDK...
09 Sep 2021 11:46:32,993 [INFO] (gamelift.js) 220: GameLiftServerAPI.InitSDK succeeded
09 Sep 2021 11:46:32,993 [INFO] (gamelift.js) 223: Waiting for Realtime server to
start...
09 Sep 2021 11:46:33,15 [WARN] (index.js) 204: Connection is INSECURE. Messages will be
sent/received as plaintext.
```

存取伺服器日誌

當遊戲工作階段結束時，Amazon GameLift Servers會自動將日誌存放在 Amazon S3 中，並保留 14 天。您可以使用 [GetGameSessionLogUrl API 呼叫](#) 來取得遊戲工作階段日誌的位置。使用 API 呼叫傳回的 URL 下載日誌。

調整記錄層級

日誌可以快速成長並超過大小限制。您應該監控日誌，並將日誌輸出限制為僅必要訊息。對於 Amazon GameLift ServersRealtime，您可以在機群的執行期組態中以形式提供參數來調整記錄層級 `loggingLevel: LOGGING_LEVEL`，其中 **LOGGING_LEVEL** 是下列其中一個值：

1. debug
2. info (預設)
3. warn
4. error
5. fatal
6. cxfatal

此清單的順序是從最不嚴重 (debug) 到最嚴重 (cxfatal)。您設定單一 `loggingLevel`，伺服器只會記錄該嚴重性層級或更高嚴重性層級的訊息。例如，設定 `loggingLevel:error` 會使機群中的所有伺服器僅將 `error`、`fatal` 和 `cxfatal` 訊息寫入日誌。

您可以在建立機群時或之後設定機群的記錄層級。在機群執行後變更其記錄層級，只會影響更新後所建立遊戲工作階段的日誌。任何現有遊戲工作階段的日誌都不會受到影響。如果您在建立機群時未設定記錄層級，您的伺服器預設會將記錄層級設定為 `info`。如需設定記錄層級的說明，請參閱下列各節。

建立Amazon GameLift ServersRealtime機群時設定記錄層級（主控台）

遵循 中的指示[建立Amazon GameLift Servers受管 EC2 機群](#)來建立您的機群，並新增以下內容：

- 在程序管理步驟的伺服器程序配置子步驟中，提供記錄層級索引鍵/值對（例如 `loggingLevel:error`）做為啟動參數的值。使用非英數字元（逗號除外）將記錄層級與任何其他參數（例如 `loggingLevel:error +map Winter444`）分隔。

建立Amazon GameLift ServersRealtime機群時設定記錄層級 (AWS CLI)

遵循 中的指示[建立Amazon GameLift Servers受管 EC2 機群](#)來建立您的機群，並新增以下內容：

- 在 `--runtime-configuration` 參數的 引數中[create-fleet](#)，提供記錄層級鍵值對（例如 `loggingLevel:error`）做為 的值Parameters。使用非英數字元（逗號除外）分隔記錄層級與任何其他參數。請參閱下列範例：

```
--runtime-configuration "GameSessionActivationTimeoutSeconds=60,  
                          MaxConcurrentGameSessionActivations=2,  
                          ServerProcesses=[{LaunchPath=/local/game/myRealtimeLaunchScript.js,  
                                              Parameters=loggingLevel:error +map Winter444,  
                                              ConcurrentExecutions=10}]"
```

設定執行中Amazon GameLift ServersRealtime機群的記錄層級（主控台）

遵循 中的指示[更新Amazon GameLift Servers機群組態](#)，使用Amazon GameLift Servers主控台更新您的機群，並新增以下內容：

- 在編輯機群頁面的伺服器程序配置下，提供記錄層級索引鍵/值對（例如 `loggingLevel:error`）做為啟動參數的值。使用非英數字元（逗號除外）將記錄層級與任何其他參數（例如 `loggingLevel:error +map Winter444`）分隔。

設定執行中Amazon GameLift ServersRealtime機群的記錄層級 (AWS CLI)

遵循 中的指示[更新Amazon GameLift Servers機群組態](#)，使用 更新您的機群 AWS CLI，並新增以下內容：

- 在 參數`--runtime-configuration`的 引數中[update-runtime-configuration](#)，提供記錄層級鍵值對（例如 `loggingLevel:error`）做為 的值Parameters。使用非英數字元（逗號除外）分隔記錄層級與任何其他參數。請參閱下列範例：

```
--runtime-configuration "GameSessionActivationTimeoutSeconds=60,  
    MaxConcurrentGameSessionActivations=2,  
    ServerProcesses=[{LaunchPath=/local/game/myRealtimeLaunchScript.js,  
        Parameters=loggingLevel:error +map Winter444,  
        ConcurrentExecutions=10}]"
```

Amazon GameLift Servers 中的安全性

如果您使用 Amazon GameLift ServersFleetIQ作為 Amazon EC2 的獨立功能，請參閱《[Amazon EC2 使用者指南](#)》中的 [Amazon EC2 中的安全性](#)。Amazon EC2

的雲端安全 AWS 是最高優先順序。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全性是 AWS 與您之間共同責任。[共同責任模型](#)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 Cloud AWS 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。若要了解適用於的合規計劃Amazon GameLift Servers，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。您也必須負責其他因素，包括資料的機密性、您公司的要求，以及適用的 I AWS 和法規。

本文件有助於您了解如何在使用 Amazon GameLift Servers 時套用共同責任模型。下列主題說明如何將 Amazon GameLift Servers 設定為達到您的安全及法規遵循目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon GameLift Servers 資源。

主題

- [Amazon GameLift Servers 中的資料保護](#)
- [適用於 Amazon GameLift Servers 的 Identity and Access Management](#)
- [使用 記錄和監控 Amazon GameLift Servers](#)
- [Amazon GameLift Servers 的合規驗證](#)
- [Amazon GameLift Servers 中的恢復能力](#)
- [Amazon GameLift Servers 中的基礎設施安全](#)
- [Amazon GameLift Servers 中的組態與漏洞分析](#)
- [Amazon GameLift Servers 的安全最佳實務](#)

Amazon GameLift Servers 中的資料保護

如果您使用 Amazon GameLift ServersFleetIQ做為 Amazon EC2 的獨立功能，請參閱《[Amazon EC2 使用者指南](#)》中的 [Amazon EC2 中的安全性](#)。Amazon EC2

AWS [共同責任模型](#)適用於 中的資料保護Amazon GameLift Servers。如此模型所述，AWS 負責保護執行所有的 全球基礎設施 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 共同的責任模型和 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 GameLift Servers 或使用主控台、API AWS CLI或其他 AWS 服務 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。

Amazon GameLift Servers 特定資料的處理方式如下：

- 您上傳到的遊戲伺服器建置和指令碼Amazon GameLift Servers會存放在 Amazon S3 中。上傳這些資料後，客戶就無法直接存取這些資料。授權使用者可以暫時存取上傳檔案，但無法直接檢視或更新 Amazon S3 中的檔案。若要刪除指令碼和組建，請使用 Amazon GameLift Servers 主控台或服務 API。
- 遊戲工作階段日誌資料會在遊戲工作階段完成後，在 Amazon S3 中存放一段時間。授權使用者可透過 Amazon GameLift Servers 主控台或呼叫服務 API 來存取日誌資料。
- 指標和事件資料會存放在 Amazon GameLift Servers 中，並可透過 Amazon GameLift Servers 主控台或呼叫服務 API 來存取。您可以在機群、執行個體、遊戲工作階段置放、配對票、遊戲工作階段

段和玩家工作階段上擷取資料。您也可以透過 Amazon CloudWatch 和 CloudWatch Events 存取資料。

- 客戶提供的資料會存放在 Amazon GameLift Servers 中。授權使用者可以透過呼叫服務 API 來存取它。潛在的敏感資料可能包括玩家資料、玩家工作階段和遊戲工作階段資料 (包括連線資訊)、配對系統資料等。

Note

如果您在請求中提供自訂玩家 ID，則會預期這些數值為匿名 UUID，且不包含任何識別玩家資訊。

如需資料保護的詳細資訊，請參閱安全部落格上的[AWS 共同責任模型和 GDPR](#) AWS 部落格文章。

靜態加密

Amazon GameLift Servers 特定資料的靜態加密處理方式如下：

- 遊戲伺服器建置和指令碼存放在具有伺服器端加密的 Amazon S3 儲存貯體中。
- 客戶提供的資料以加密的格式存放在 Amazon GameLift Servers 中。

傳輸中加密

與 Amazon GameLift Servers APIs 的連線是透過安全 (SSL) 連線進行，並使用 [AWS Signature 第 4 版](#) 進行驗證 (透過 CLI 或 AWS SDK AWS 連線時，會自動處理簽署)。身分驗證是使用安全登入資料 (用來建立連線) 的 IAM 定義的存取政策來管理。

遊戲客戶端與遊戲伺服器之間的直接通訊如下：

- 對於在 Amazon GameLift Servers 資源上託管的自訂遊戲伺服器，通訊不涉及 Amazon GameLift Servers 服務。客戶須負責此通訊的加密。您可以使用啟用 TLS 的叢集，讓遊戲用戶端在連線時驗證遊戲伺服器，並加密遊戲用戶端與遊戲伺服器之間的所有通訊。
- 對於啟用 TLS 憑證產生 Amazon GameLift Servers Realtime 功能的，使用適用於的用戶端 SDK 的遊戲用戶端與 Realtime 伺服器之間的流量 Realtime 會在傳輸中加密。TCP 流量會使用 TLS 1.2 加密，而 UDP 流量則使用 DTLS 1.2 加密。

網際網路流量隱私權

您可以安全地從遠端存取 Amazon GameLift Servers 執行個體。對於使用 Linux 的執行個體，SSH 會為遠端存取提供安全的通訊頻道。對於執行 Windows 的執行個體，請使用遠端桌面通訊協定 (RDP) 用戶端。使用時 Amazon GameLift Servers FleetIQ，使用 AWS Systems Manager Session Manager 和 Run Command 對執行個體的遠端存取會使用 TLS 1.2 加密，並使用 SigV4 簽署建立連線的請求。如需連接至受管 Amazon GameLift Servers 執行個體的說明，請參閱[遠端連線至 Amazon GameLift Servers 機群執行個體](#)。

適用於 Amazon GameLift Servers 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 管理員可以控制身分驗證 (已登入) 和授權 (具有許可) 以使用 Amazon GameLift Servers 資源。IAM 是您可以免費使用 AWS 服務的。

主題

- [目標對象](#)
- [使用身分驗證](#)
- [使用政策管理存取權](#)
- [Amazon GameLift Servers 搭配 IAM 的運作方式](#)
- [Amazon GameLift Servers 的身分型政策範例](#)
- [對 Amazon GameLift Servers 身分與存取進行疑難排解](#)
- [AWS 的受管政策 Amazon GameLift Servers](#)

目標對象

使用方式 AWS Identity and Access Management (IAM) 會有所不同，取決於您在 中執行的工作 Amazon GameLift Servers。

服務使用者：如果使用 Amazon GameLift Servers 執行任務，管理員會為您提供所需的憑證和許可。隨著您為了執行作業而使用的 Amazon GameLift Servers 功能數量變多，您可能會需要額外的許可。了解存取的管理方式可協助您向管理員請求正確的許可。若您無法存取 Amazon GameLift Servers 中的某項功能，請參閱 [對 Amazon GameLift Servers 身分與存取進行疑難排解](#)。

服務管理員：如果您負責公司內的 Amazon GameLift Servers 資源，您可能具備 Amazon GameLift Servers 的完整存取權限。您的任務是判斷服務使用者應存取的 Amazon GameLift Servers 功能及資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Amazon GameLift Servers 使用 IAM 的方式，請參閱 [Amazon GameLift Servers 搭配 IAM 的運作方式](#)。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 Amazon GameLift Servers 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的範例 Amazon GameLift Servers 身分型政策，請參閱 [Amazon GameLift Servers 的身分型政策範例](#)。

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色來驗證（登入 AWS）。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分登入。AWS IAM Identity Center (IAM Identity Center) 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱 AWS 登入《使用者指南》中的 [如何登入您的 AWS 帳戶](#)。

如果您以 AWS 程式設計方式存取，AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的 [適用於 API 請求的 AWS Signature 第 4 版](#)。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的 [多重要素驗證](#) 和《IAM 使用者指南》中的 [IAM 中的 AWS 多重要素驗證](#)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取帳戶中的所有 AWS 服務和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的 [需要根使用者憑證的任務](#)。

聯合身分

根據最佳實務，要求人類使用者，包括需要管理員存取權的使用者，使用身分提供者的聯合身分來 AWS 服務 使用臨時憑證來存取。

聯合身分是您企業使用者目錄、Web 身分提供者、AWS Directory Service、Identity Center 目錄或任何使用透過身分來源提供的登入資料 AWS 服務 存取的使用者。當聯合身分存取時 AWS 帳戶，它們會擔任 角色，而角色會提供臨時登入資料。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連接並同步到您自己的身分來源中的一組使用者 AWS 帳戶 和群組，以便在所有 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的[什麼是 IAM Identity Center？](#)。

IAM 使用者和群組

[IAM 使用者](#)是 中的身分 AWS 帳戶，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 [IAM 使用者指南](#)中的為需要長期憑證的使用案例定期輪換存取金鑰。

[IAM 群組](#)是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的[IAM 使用者的使用案例](#)。

IAM 角色

[IAM 角色](#)是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從[使用者切換至 IAM 角色（主控台）](#)。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的[擔任角色的方法](#)。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊

訊，請參閱《[IAM 使用者指南](#)》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的[許可集](#)。

- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以直接將政策連接到資源 (而不是使用角色做為代理)。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的[IAM 中的跨帳戶資源存取](#)。
- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
- 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱[《轉發存取工作階段》](#)。
- 服務角色 – 服務角色是服務擔任的 [IAM 角色](#)，可代表您執行動作。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權給 AWS 服務](#)。
- 服務連結角色 – 服務連結角色是一種連結至 的服務角色類型 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 中 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。
- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式](#)。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件，當與身分或資源相關聯時，AWS 會定義其許可。當委託人 (使用者、根使用者或角色工作階段) 發出

請求時，會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 的形式存放在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 [JSON 政策概觀](#)。

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。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

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](#)。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，請參閱《IAM 使用者指南》中的 [在受管政策和內嵌政策間選擇](#)。

資源型政策

資源型政策是連接到資源的 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。

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

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。

Amazon S3 AWS WAF 和 Amazon VPC 是支援 ACLs 的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的 [存取控制清單 \(ACL\) 概觀](#)。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- **許可界限** – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 Principal 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 [IAM 實體許可界限](#)。
- **服務控制政策 (SCPs)** – SCPs 是 JSON 政策，可指定中組織或組織單位 (OU) 的最大許可 AWS Organizations。AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶之多個的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [服務控制政策](#)。
- **資源控制政策 (RCP)** – RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs 的詳細資訊，包括支援 RCPs AWS 服務的清單，請參閱 AWS Organizations 《使用者指南》中的 [資源控制政策 \(RCPs\)](#)。
- **工作階段政策** – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的 [工作階段政策](#)。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多個政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的 [政策評估邏輯](#)。

Amazon GameLift Servers 搭配 IAM 的運作方式

在您使用 IAM 管理 Amazon GameLift Servers 的存取權之前，請了解搭配 Amazon GameLift Servers 使用的 IAM 功能有哪些。

您可搭配 Amazon GameLift Servers 使用的 IAM 功能

IAM 功能	Amazon GameLift Servers 支援
身分型政策	是
資源型政策	否
政策動作	是
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)	是
臨時憑證	是
主體許可	是
服務角色	是
服務連結角色	否

若要全面了解 Amazon GameLift Servers 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《IAM 使用者指南》中的與 IAM [AWS 搭配使用的 服務](#)。

適用於 Amazon GameLift Servers 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。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。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 GameLift Servers 的身分型政策範例

若要檢視 Amazon GameLift Servers 身分型政策範例，請參閱 [Amazon GameLift Servers 的身分型政策範例](#)。

Amazon GameLift Servers 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，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，做為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當委託人和資源位於不同位置時 AWS 帳戶，信任帳戶中的 IAM 管理員也必須授予委託人實體（使用者或角色）存取資源的許可。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。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](#)。

適用於 Amazon GameLift Servers 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。政策動作通常具有與相關聯 AWS API 操作相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

如需 Amazon GameLift Servers 動作清單，請參閱《服務授權參考》中的 [定義的動作 Amazon GameLift Servers](#)。

Amazon GameLift Servers 中的政策動作會在動作之前使用以下字首：

```
gamelift
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [  
    "gamelift:action1",  
    "gamelift:action2"  
]
```

您也可以使用萬用字元 (*) 來指定多個動作。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": "gamelift:Describe*"
```

若要檢視 Amazon GameLift Servers 身分型政策範例，請參閱 [Amazon GameLift Servers 的身分型政策範例](#)。

Amazon GameLift Servers 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。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

如需 Amazon GameLift Servers 資源類型及其 ARNs，請參閱《服務授權參考》中的 [定義的資源 Amazon GameLift Servers](#)。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 GameLift Servers 定義的動作](#)。

有些 Amazon GameLift Servers 資源具有 ARN 值，可讓資源使用 IAM 政策來管理其存取。Amazon GameLift Servers 機群資源具有下列語法的 ARN：

```
arn:${Partition}:gamelift:${Region}:${Account}:fleet/${FleetId}
```

如需有關 ARN 格式的詳細資訊，請參閱《AWS 一般參考》中的 [Amazon Resource Name \(ARN\)](#)。

例如，若要在陳述式中指定 fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa 機群，請使用以下 ARN。

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

若要指定屬於特定帳戶的所有機群，請使用萬用字元 (*)：

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/*"
```

若要檢視 Amazon GameLift Servers 身分型政策範例，請參閱 [Amazon GameLift Servers 的身分型政策範例](#)。

Amazon GameLift Servers 的政策條件索引鍵

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 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。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。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵，AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，會使用邏輯 OR 操作 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 [IAM 政策元素：變數和標籤](#)。

AWS 支援全域條件金鑰和服務特定的條件金鑰。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](#)。

如需 Amazon GameLift Servers 條件索引鍵的清單，請參閱《服務授權參考》中的 [的條件索引鍵 Amazon GameLift Servers](#)。若要了解您可以針對何種動作及資源使用條件索引鍵，請參閱 [Amazon GameLift Servers 定義的動作](#)。

若要檢視 Amazon GameLift Servers 身分型政策範例，請參閱 [Amazon GameLift Servers 的身分型政策範例](#)。

Amazon GameLift Servers 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 GameLift Servers

支援 ABAC (政策中的標籤)：是

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在 AWS 中，這些屬性稱為標籤。您可以將標籤連接至 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

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 \(ABAC\)](#)。

如需根據資源上的標籤限制資源存取的身分型政策範例，請參閱 [根據標籤檢視 Amazon GameLift Servers 機群](#)。

將臨時憑證與 Amazon GameLift Servers 搭配使用

支援臨時憑證：是

當您使用臨時登入資料登入時，有些 AWS 服務 無法運作。如需詳細資訊，包括哪些 AWS 服務 使用臨時登入資料，請參閱 [《AWS 服務 IAM 使用者指南》中的 使用 IAM](#)。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入，則會使用暫時登入資料。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立臨時登入資料。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的[從使用者切換至 IAM 角色 \(主控台\)](#)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。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 中的暫時性安全憑證](#)。

Amazon GameLift Servers 的跨服務主體權限

支援轉寄存取工作階段 (FAS)：是

當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱[轉發存取工作階段](#)。

Amazon GameLift Servers 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 服務服務](#)。

Warning

變更服務角色的許可有可能會讓 Amazon GameLift Servers 功能出現故障。只有 Amazon GameLift Servers 提供指引時，才能編輯服務角色。

允許 Amazon GameLift Servers 託管的遊戲伺服器存取其他 AWS 資源，例如 AWS Lambda 函數或 Amazon DynamoDB 資料庫。由於遊戲伺服器託管在 Amazon GameLift Servers 管理的機群上，因此您需要一個服務角色，以提供對其他 AWS 資源的 Amazon GameLift Servers 有限存取權。如需詳細資訊，請參閱[與機群中的其他 AWS 資源通訊](#)。

Amazon GameLift Servers 的服務連結角色

支援服務連結角色：否

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 中 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。

如需建立或管理服务連結角色的詳細資訊，請參閱《[AWS IAM 使用者指南](#)》中的[使用 IAM 的服務](#)。在資料表中尋找服務，其中包含服務連結角色欄中 Yes 的。選擇是以檢視該服務的服務連結角色文件。

Amazon GameLift Servers 的身分型政策範例

根據預設，使用者和角色不具備建立或修改 Amazon GameLift Servers 資源的權限。他們也無法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 來執行任務。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的[建立 IAM 政策 \(主控台\)](#)。

如需 Amazon GameLift Servers 所定義之動作和資源類型的詳細資訊，包括各個資源類型的 ARN 格式，請參閱《服務授權參考》中的《[適用於 Amazon GameLift Servers 的動作、資源和條件索引鍵](#)》。

主題

- [政策最佳實務](#)
- [使用 Amazon GameLift Servers 主控台](#)
- [允許使用者檢視他們自己的許可](#)
- [允許玩家存取遊戲工作階段](#)
- [允許存取一個 Amazon GameLift Servers 佇列](#)
- [根據標籤檢視 Amazon GameLift Servers 機群](#)
- [在 Amazon S3 中存取遊戲組建檔案](#)

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amazon GameLift Servers 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 [AWS 受管政策](#) 或 [任務職能的 AWS 受管政策](#)。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 [IAM 中的政策和許可](#)。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 [IAM JSON 政策元素：條件](#)。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [使用 IAM Access Analyzer 驗證政策](#)。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html 中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 [IAM 安全最佳實務](#)。

使用 Amazon GameLift Servers 主控台

若要存取 Amazon GameLift Servers 主控台，您必須擁有最低的一組許可。這些許可必須允許您列出和檢視 中 Amazon GameLift Servers 資源的詳細資訊 AWS 帳戶。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

為了確保這些實體仍然可以使用 Amazon GameLift Servers 主控台，請在下列範例和 中使用 語法將許可新增至使用者和群組 [管理許可範例](#)。如需詳細資訊，請參閱 [設定 的使用者許可 Amazon GameLift Servers](#)。

透過 Amazon GameLift Servers AWS CLI 或 AWS API 操作使用 的使用者不需要最低主控台許可。反之，您可以限制僅存取使用者需要執行的操作。例如，代表遊戲用戶端的玩家使用者需要請求遊戲工作階段、將玩家放入遊戲和其他任務的存取權。

如需使用所有 Amazon GameLift Servers 主控台功能所需的許可資訊，請參閱 中管理員的許可語法 [管理許可範例](#)。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台或使用或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

允許玩家存取遊戲工作階段

若要將玩家放入遊戲工作階段，遊戲用戶端和後端服務需要許可。如需這些案例的政策範例，請參閱[玩家使用者許可範例](#)。

允許存取一個Amazon GameLift Servers佇列

下列範例提供使用者存取特定Amazon GameLift Servers佇列的權限。

此政策授予使用者使用下列動作新增、更新和刪除佇列目的地的許

可：gamelift:UpdateGameSessionQueue、gamelift>DeleteGameSessionQueue和gamelift:DescribeGameSessionQueues。如圖所示，此政策使用 Resource元素來限制對單一佇列的存取：gamesessionqueue/examplequeue123。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewSpecificQueueInfo",
      "Effect": "Allow",
      "Action": [
        "gamelift:DescribeGameSessionQueues"
      ],
      "Resource": "arn:aws:gamelift:us-east-1:555555555555:gamesessionqueue/examplequeue123"
    },
    {
      "Sid": "ManageSpecificQueue",
      "Effect": "Allow",
      "Action": [
        "gamelift:UpdateGameSessionQueue",
        "gamelift>DeleteGameSessionQueue"
      ],
      "Resource": "arn:aws:gamelift:us-east-1:111122223333:gamesessionqueue/examplequeue123"
    }
  ]
}
```

根據標籤檢視 Amazon GameLift Servers 機群

您可以在身分型政策中使用條件，以根據標籤控制存取 Amazon GameLift Servers 資源。此範例示範如何建立政策，允許在Owner標籤符合使用者的使用者名稱時檢視機群。此政策也會授予在 主控台中完成此操作所需的許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListFleetsInConsole",
      "Effect": "Allow",
      "Action": "gamelift:ListFleets",
      "Resource": "*"
    },
    {
      "Sid": "ViewFleetIfOwner",
      "Effect": "Allow",
      "Action": "gamelift:DescribeFleetAttributes",
      "Resource": "arn:aws:gamelift:*:*:fleet/*",
      "Condition": {
        "StringEquals": {"aws:ResourceAccount": "${aws:username}"}
      }
    }
  ]
}
```

在 Amazon S3 中存取遊戲組建檔案

將遊戲伺服器與 整合之後Amazon GameLift Servers，請將建置檔案上傳至 Amazon S3。若要 Amazon GameLift Servers讓 存取建置檔案，請使用下列政策。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion"
        ],
        "Resource": "arn:aws:s3:::bucket-name/object-name"
    }
}
```

如需上傳 Amazon GameLift Servers 遊戲檔案的詳細資訊，請參閱 [部署用於 Amazon GameLift Servers 託管的自訂伺服器建置](#)。

對 Amazon GameLift Servers 身分與存取進行疑難排解

使用以下資訊來協助您診斷和修正使用 Amazon GameLift Servers 和 AWS Identity and Access Management (IAM) 時可能遇到的常見問題。

主題

- [我未獲授權，不得在 Amazon GameLift Servers 中執行動作](#)
- [我未獲得執行 iam:PassRole 的授權](#)
- [我想要允許以外的人員 AWS 帳戶 存取我的 Amazon GameLift Servers 資源](#)

我未獲授權，不得在 Amazon GameLift Servers 中執行動作

如果 AWS Management Console 告知您無權執行 動作，請聯絡您的 AWS 帳戶管理員尋求協助。您的管理員是為您提供簽署憑證的人員。

當 IAM mateojackson 使用者嘗試使用主控台檢視佇列的詳細資訊，但沒有 gamelift:DescribeGameSessionQueues 許可時，會發生下列範例錯誤：

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
gamelift:DescribeGameSessionQueues on resource: examplequeue123
```

在此情況下，Mateo 會要求管理員更新其政策，以允許他使用 gamelift:DescribeGameSessionQueues 動作來讀取 examplequeue123 資源的存取權。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您未獲授權執行 iam:PassRole 動作，您的政策必須更新，允許您將角色傳遞給 Amazon GameLift Servers。

有些 AWS 服務可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

名為 marymajor 的 IAM 使用者嘗試使用主控台在 Amazon GameLift Servers 中執行動作時，發生下列範例錯誤。但是，動作要求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許以外的人員 AWS 帳戶 存取我的 Amazon GameLift Servers 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon GameLift Servers 是否支援這些功能，請參閱 [Amazon GameLift Servers 搭配 IAM 的運作方式](#)。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱《[IAM 使用者指南](#)》中的 [在您擁有 AWS 帳戶 的另一個 中為 IAM 使用者提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的 [將存取權提供給第三方 AWS 帳戶 擁有](#)。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的 [將存取權提供給在外部進行身分驗證的使用者 \(聯合身分\)](#)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 [IAM 中的跨帳戶資源存取](#)。

AWS 的 受管政策 Amazon GameLift Servers

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。我們建議您定義使用案例專屬的[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主體身分（使用者、群組和角色）。當新的 AWS 服務 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](#)。

AWS 受管政策：GameLiftContainerFleetPolicy

您可以GameLiftContainerFleetPolicy連接到 IAM 角色。

政策會授予Amazon GameLift Servers容器機群中運算動作的許可。容器機群是一組為您Amazon GameLift Servers管理的託管資源。Amazon GameLift Servers需要代表您連線到 Amazon GameLift Servers服務和其他 AWS 服務的許可。

使用 建立容器機群時Amazon GameLift Servers，請提供已連接 GameLiftContainerFleetPolicy 受管政策的 IAM 服務角色。如需建立服務角色的說明，請參閱 [設定的 IAM 服務角色 Amazon GameLift Servers](#)。

如需詳細資訊，請參閱 [GameLiftContainerFleetPolicy](#)。

許可詳細資訊

此政策包含以下許可。

- cloudwatch – 允許 Amazon GameLift Servers將遊戲工作階段日誌寫入您 AWS 帳戶中的 Amazon CloudWatch Events 日誌串流。
- cloudwatch – 允許 Amazon GameLift Servers建立 CloudWatch 日誌群組，以在日誌串流中組織遊戲工作階段資料。
- s3 – 允許 Amazon GameLift Servers 將遊戲工作階段日誌寫入您 AWS 帳戶中的 Amazon Simple Storage Service 儲存貯體。
- s3 – 允許 使用 API 動作 Amazon GameLift Servers擷取 AWS 區域 指定 Amazon S3 儲存貯體所在的 s3:GetBucketLocation。

- gamelift – 允許 Amazon GameLift Servers擷取身分驗證字符，以允許託管遊戲伺服器Amazon GameLift Servers透過您的帳戶與服務通訊 AWS。

Amazon GameLift ServersAWS 受管政策的更新

檢視自此服務開始追蹤這些變更Amazon GameLift Servers以來，AWS 受管政策更新的詳細資訊。如需此頁面變更的自動提醒，請訂閱[Amazon GameLift Servers版本備註](#)頁面上的 RSS 摘要。

變更	描述	日期
GameLiftContainerFleetPolicy – 變更	Amazon GameLift Servers 新增了擷取 Amazon S3 儲存貯 AWS 區域 體 的新許可。	2024 年 2 月 5 日
GameLiftContainerFleetPolicy – 新政策	Amazon GameLift Servers 新增了許可，讓遊戲伺服器容器能夠在Amazon GameLift Servers受管機群上執行。	2024 年 11 月 12 日
Amazon GameLift Servers 已開始追蹤變更	Amazon GameLift Servers 已開始追蹤其 AWS 受管政策的變更。	2024 年 11 月 12 日

使用 記錄和監控 Amazon GameLift Servers

監控是維護 和 AWS 解決方案的可靠性、可用性Amazon GameLift Servers和效能的重要部分。您應該從 AWS 解決方案的所有部分收集監控資料，以便在發生多點失敗時更輕鬆地偵錯。

AWS 和 Amazon GameLift Servers提供數種工具來監控您的遊戲託管資源，並回應潛在的事件。

Amazon CloudWatch 警示

使用 Amazon CloudWatch 警示，您可在自己指定的一段時間內監看單一指標。如果指標超過指定的閾值，則會將通知傳送至 Amazon SNS 主題或 AWS Auto Scaling 政策。CloudWatch 警示會在其狀態變更時觸發，並且會維持一段指定的期間，而不是處於特定狀態。如需詳細資訊，請參閱[使用 Amazon CloudWatch 監控 Amazon GameLift Servers](#)。

AWS CloudTrail 日誌

CloudTrail 提供使用者、角色或 AWS 服務在 中採取之動作的記錄 Amazon GameLift Servers。您可以利用 CloudTrail 所收集的資訊來判斷向 Amazon GameLift Servers 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。如需詳細資訊，請參閱[使用 記錄 Amazon GameLift Servers API 呼叫 AWS CloudTrail](#)。

Amazon GameLift Servers 的合規驗證

Amazon GameLift Servers 不在任何 AWS 合規計劃的範圍內。

若要了解 是否 AWS 服務 在特定合規計劃範圍內，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 中的 AWS Artifact](#)報告。

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。AWS 提供下列資源以協助合規：

- [安全合規與治理](#) - 這些解決方案實作指南內容討論了架構考量，並提供部署安全與合規功能的步驟。
- [HIPAA 合格服務參考](#) - 列出 HIPAA 合格服務。並非所有 AWS 服務 都符合 HIPAA 資格。
- [AWS 合規資源](#) - 此工作手冊和指南集合可能適用於您的產業和位置。
- [AWS 客戶合規指南](#) - 透過合規的角度了解共同責任模型。本指南摘要說明跨多個架構（包括國家標準與技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 保護 AWS 服務 並映射指引至安全控制的最佳實務。
- 《AWS Config 開發人員指南》中的[使用 規則評估資源](#) - AWS Config 服務會評估資源組態符合內部實務、產業準則和法規的程度。
- [AWS Security Hub](#) - 這 AWS 服務 可讓您全面檢視其中的安全狀態 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱「[Security Hub 控制參考](#)」。
- [Amazon GuardDuty](#) - 這會監控您的環境是否有可疑和惡意活動，以 AWS 服務 偵測對您 AWS 帳戶、工作負載、容器和資料的潛在威脅。GuardDuty 可滿足特定合規架構所規定的入侵偵測需求，以協助您因應 PCI DSS 等各種不同的合規需求。
- [AWS Audit Manager](#) - 這 AWS 服務 可協助您持續稽核 AWS 用量，以簡化您管理風險的方式，以及是否符合法規和業界標準。

Amazon GameLift Servers 中的恢復能力

如果您使用 Amazon GameLift Servers FleetIQ 作為 Amazon EC2 的獨立功能，請參閱 [《Amazon EC2 使用者指南》中的 Amazon EC2 中的安全性](#)。Amazon EC2

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置的。AWS 區域提供多個實體隔離和隔離的可用區域，這些區域以低延遲、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](#)。

除了 AWS 全球基礎設施之外，Amazon GameLift Servers 還提供下列功能，以協助支援您的資料彈性需求：

- 多區域佇列 – Amazon GameLift Servers 遊戲工作階段佇列用於放置具有可用託管資源的新遊戲工作階段。跨越多個區域的佇列能夠在區域發生服務中斷時重新導向遊戲工作階段置放。如需建立遊戲工作階段佇列的詳細資訊和最佳實務，請參閱 [自訂遊戲工作階段佇列](#)。
- 自動容量擴展 – 使用 Amazon GameLift Servers 擴展工具維持託管資源的運作狀態和可用性。這些工具提供一系列的選項，可讓您根據遊戲和玩家的需求調整機群容量。如需擴展的詳細資訊，請參閱 [使用擴展遊戲託管容量 Amazon GameLift Servers](#)。
- 分佈於各個執行個體 – 根據機群大小，將傳入流量 Amazon GameLift Servers 分佈於多個執行個體。最佳實務是，生產環境中的遊戲應該有多個執行個體，以便在執行個體狀況不良或沒有回應時維持可用性。
- Amazon S3 儲存 – 上傳至的遊戲伺服器建置和指令碼會使用標準儲存類別 Amazon GameLift Servers 儲存在 Amazon S3 中，該類別使用多個資料中心複寫來提高彈性。遊戲工作階段日誌也會使用標準儲存類別儲存在 Amazon S3 中。

Amazon GameLift Servers 中的基礎設施安全

如果您使用 Amazon GameLift Servers FleetIQ 作為 Amazon EC2 的獨立功能，請參閱 [《Amazon EC2 使用者指南》中的 Amazon EC2 中的安全性](#)。Amazon EC2

作為受管服務，Amazon GameLift Servers 受到 [Amazon Web Services：安全程序概觀](#) 白皮書中所述的 AWS 全球網路安全程序的保護。

您可以使用 AWS 已發佈的 API 呼叫，Amazon GameLift Servers 透過網路存取。用戶端必須支援 Transport Layer Security (TLS) 1.2 或更新版本。建議使用 TLS 1.3 或更新版本。用戶端也必須支援

具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 [AWS Security Token Service](#) (AWS STS) 來產生暫時安全憑證來簽署請求。

Amazon GameLift Servers 此服務會將所有機群放入 Amazon Virtual Private Cloud (VPCs)，讓每個機群都存在於 AWS 雲端的邏輯隔離區域中。您也可以使用 Amazon GameLift Servers 政策，從特定 VPC 端點或特定 VPC 控制存取。實際上，這只會隔離 AWS 網路內特定 VPC 對 Amazon GameLift Servers 資源的網路存取。建立機群時，請指定連接埠號碼和 IP 地址的範圍。這些範圍會限制輸入流量如何能夠存取機群 VPC 上的託管遊戲伺服器。選擇機群存取設定時，請使用標準安全最佳實務。

Amazon GameLift Servers 中的組態與漏洞分析

如果您使用 Amazon GameLift Servers FleetIQ 作為 Amazon EC2 的獨立功能，請參閱 [《Amazon EC2 使用者指南》中的 Amazon EC2 中的安全性](#)。Amazon EC2

組態和 IT 控制是 AWS 與身為我們客戶的您共同的責任。如需詳細資訊，請參閱 AWS [共同責任模型](#)。AWS 會處理基本安全任務，例如訪客作業系統 (OS) 和資料庫修補、防火牆組態和災難復原。這些程序已由適當的第三方進行檢閱並認證。如需詳細資訊，請參閱下列資源：[Amazon Web Services：安全程序概觀](#)（白皮書）。

以下安全最佳實務也可處理 Amazon GameLift Servers 中的組態和漏洞分析：

- 客戶必須負責管理部署到 Amazon GameLift Servers 執行個體以進行遊戲託管的軟體。具體而言：
 - 應維護客戶提供的遊戲伺服器應用程式軟體，包括更新和安全修補程式。若要更新遊戲伺服器軟體，請將新組建上傳到 Amazon GameLift Servers 或在新的容器映像中提供，為其建立新的機群，並將流量重新導向至新的機群。
 - 基本 Amazon Machine Image (AMI)，其中包括作業系統，只有在建立新機群時才會更新。若要修補、更新和保護屬於 AMI 的作業系統和其他應用程式，請定期取代機群，無論遊戲伺服器更新為何。至少每年應替換一次生產機群。如需詳細資訊，請參閱[讓機群執行時間環境保持在最新狀態](#)。
- 客戶應考慮使用最新的 SDK 版本定期更新其遊戲，包括 AWS SDK、Amazon GameLift Servers 伺服器 SDK 和適用於 Realtime Server 的 Amazon GameLift Servers 用戶端 SDK。

Amazon GameLift Servers 的安全最佳實務

如果您使用 Amazon GameLift Servers FleetIQ 作為 Amazon EC2 的獨立功能，請參閱 [《Amazon EC2 使用者指南》中的 Amazon EC2 中的安全性](#)。Amazon EC2

在您開發和實作自己的安全政策時，可考慮使用 Amazon GameLift Servers 提供的多種安全功能。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

讓機群執行時間環境保持在最新狀態

Amazon GameLift Servers 強烈建議您定期取代受管機群（包括受管 EC2 和受管容器機群），以維護遊戲伺服器的安全執行期環境。在沒有執行時間更新的情況下長時間執行的機群可能包含過時的相依性和可能危害您遊戲伺服器的安全漏洞。如需如何為部署到機群Amazon GameLift Servers的軟體分擔責任的詳細資訊，請參閱 [Amazon GameLift Servers 中的組態與漏洞分析](#)。

受管機群的執行期環境取決於其 Amazon Machine Image (AMI) 版本。建立新機群時，會將最新的可用 AMI 版本Amazon GameLift Servers指派給機群，且該機群中的所有運算執行個體都會以該版本部署。若要更新 AMI 版本，您必須建立新的機群。如需目前 AMI 版本的詳細資訊，請參閱 [Amazon GameLift Servers AMI 版本](#)。

建議的實務：

- 監控機群存留期並取代超過 30 天的機群 – 您可以在Amazon GameLift Servers主控台中追蹤機群的建立日期，或使用 CLI 擷取機群屬性。會在主控台中針對超過 90 天的機群Amazon GameLift Servers顯示警告，並針對超過一年的機群透過電子郵件通知帳戶持有人。

Note

更新機群（例如使用 [UpdateFleetAttributes](#) 或 [UpdateContainerFleet](#)）不會變更 AMI 版本。您必須建立新的機群。

- 根據安全運作狀態定期取代機群 – 設定定期排程以建立新機群並淘汰舊機群。請考慮使用 [Amazon Q](#) 之類的服務，以目前 AMI 版本檢閱您的遊戲程式碼、偵測安全問題，並建議修復步驟。
- 在部署之前使用最新的 AMI 版本測試伺服器建置 – 在建立新的機群Amazon GameLift Servers之前，您可能需要修改伺服器建置並將其上傳至。
- 管理 AWS 帳戶的機群配額 – 如果需要建立替代機群，您可以請求提高限制。如需詳細資訊，請參閱[Amazon GameLift Servers 端點和配額](#)。
- 考慮自動化機群替換 – 您可以自動化程序來建立新的機群，並從較舊的機群遷移玩家流量。例如：
 - 使用 AWS CloudFormation 將機群建立和管理自動化。將您的機群組態維護為 AWS CloudFormation 範本，並使用它們來啟動資源堆疊。

- 利用Amazon GameLift Servers別名功能來抽象化特定機群 IDs。機群別名可讓您輕鬆地將玩家流量從現有機群切換到新的機群，而不會中斷進行中的遊戲工作階段。如需詳細資訊，請參閱[使用別名抽象機Amazon GameLift Servers群指定](#)。
- 使用藍/綠部署策略來降低遷移風險並保持零停機時間。使用兩個相同的生產環境，您可以利用完整的類似生產的測試環境，對遷移程序進行更大的控制，並確保即時復原。

保護您的連接埠組態

我們強烈建議不要開啟網際網路的連接埠，因為這樣做會造成安全風險。例如，下列組態會開啟遠端桌面連接埠，允許網際網路上的任何人存取執行個體：

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "0.0.0.0/0",
      "Protocol": "RDP",
      "ToPort": 3389
    }
  ]
}
```

反之，請使用 [UpdateFleetPortSettings](#) 來開啟具有特定 IP 地址或地址範圍的連接埠，如本範例所示：

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "54.186.139.221/32",
      "Protocol": "TCP",
      "ToPort": 3389
    }
  ]
}
```

其他安全資源

如需如何更Amazon GameLift Servers安全地使用 的詳細資訊，請參閱 [AWS Well-Architected Tool 安全支柱](#)。

Amazon GameLift Servers 參考指南

本節包含有關使用 Amazon GameLift Servers 的參考文件。

主題

- [的服務 API Amazon GameLift Servers](#)
- [適用於 的伺服器 SDK 5.x Amazon GameLift Servers](#)
- [第 4 Amazon GameLift Servers版及更早版本的伺服器 SDK](#)
- [Amazon GameLift ServersAmazon GameLift Servers參考Realtime](#)
- [遊戲工作階段置放事件](#)
- [Amazon GameLift Servers AMI 版本](#)
- [Amazon GameLift Servers 端點和配額](#)
- [Amazon GameLift Servers API 限制](#)
- [Amazon GameLift Servers SDK API 限制](#)
- [Amazon GameLift Servers UDP ping 信標](#)

的服務 API Amazon GameLift Servers

在建置Amazon GameLift Servers遊戲託管解決方案和其他功能時，使用此任務型清單來尋找 API 操作。AWS 開發套件在 `aws.gamelift` 命名空間中包含這些操作。[下載 AWS SDK](#) 或[檢視 Amazon GameLift Servers API 參考文件](#)。您也可以使用 API 搭配 AWS 命令列界面 (AWS CLI)，如[AWS CLI 命令參考](#)中所述。

API 包含兩組受管遊戲託管的操作：

- [管理Amazon GameLift Servers託管資源](#)
- [開始遊戲工作階段並加入玩家](#)

Amazon GameLift Servers 服務 API 也包含與其他Amazon GameLift Servers工具和解決方案搭配使用的操作。如需 FleetIQ APIs 的清單，請參閱 [FleetIQ API 操作](#)。如需配對的 FlexMatch API 清單，請參閱 [FlexMatch API 操作](#)。APIs

管理Amazon GameLift Servers託管資源

呼叫這些操作來設定遊戲伺服器的託管資源、擴展容量以滿足玩家需求、取得效能和用量指標等。使用託管遊戲伺服器時，請使用這些 API 操作Amazon GameLift Servers，包括 Amazon GameLift ServersRealtime。您也可以在[Amazon GameLift Servers主控台](#)中處理大多數資源管理任務，也可以使用 AWS Command Line Interface (AWS CLI) 工具進行呼叫。

準備遊戲伺服器以進行部署

上傳並設定遊戲的遊戲伺服器程式碼，以準備在託管資源上進行部署和啟動。

管理自訂遊戲伺服器組建

- [upload-build](#) – 從本機路徑上傳建置檔案，並建立新的Amazon GameLift Servers建置資源。此操作以 AWS CLI 命令的形式提供，是上傳遊戲伺服器組建的最常見方式。
- [CreateBuild](#) – 使用存放在 Amazon S3 儲存貯體中的檔案建立新的組建。
- [ListBuilds](#) – 取得上傳至Amazon GameLift Servers區域的所有組建清單。
- [DescribeBuild](#) – 擷取與組建相關聯的資訊。
- [UpdateBuild](#) – 變更建置中繼資料，包括建置名稱和版本。
- [DeleteBuild](#) – 從 移除組建Amazon GameLift Servers。

管理Amazon GameLift ServersRealtime組態指令碼

- [CreateScript](#) – 上傳 JavaScript 檔案並建立新的Amazon GameLift Servers指令碼資源。
- [ListScripts](#) – 取得上傳至Amazon GameLift Servers區域的所有Realtime指令碼清單。
- [DescribeScript](#) – 擷取與Realtime指令碼相關聯的資訊。
- [UpdateScript](#) – 變更指令碼中繼資料並上傳修訂的指令碼內容。
- [DeleteScript](#) – 從 移除Realtime指令碼Amazon GameLift Servers。

設定用於託管的運算資源

設定託管資源，並使用遊戲伺服器建置或Realtime組態指令碼進行部署。

建立和管理機群

- [CreateFleet](#) – 設定和部署新的運算資源Amazon GameLift Servers機群，以執行您的遊戲伺服器。部署後，遊戲伺服器會依設定自動啟動，並準備好託管遊戲工作階段。

- [ListFleets](#) – 取得Amazon GameLift Servers區域中所有機群的清單。
- [DeleteFleet](#) – 移除不再執行遊戲伺服器或託管玩家的機群。
- 檢視/更新機群位置。
 - [CreateFleetLocations](#) – 將遠端位置新增至支援多個位置的現有機群
 - [DescribeFleetLocationAttributes](#) – 取得機群的所有遠端位置清單，並檢視每個位置的目前狀態。
 - [DeleteFleetLocations](#) – 從支援多個位置的機群中移除遠端位置。
- 檢視/更新機群組態。
 - [DescribeFleetAttributes](#) / [UpdateFleetAttributes](#) – 檢視或變更機群的中繼資料和設定，以保護遊戲工作階段和資源建立限制。
 - [DescribeFleetPortSettings](#) / [UpdateFleetPortSettings](#) – 檢視或變更機群允許的傳入許可 (IP 地址和連接埠設定範圍)。
 - [DescribeRuntimeConfiguration](#)/[UpdateRuntimeConfiguration](#) – 檢視或變更要在機群中每個執行個體上執行的伺服器程序 (和數量)。

管理機群容量

- [DescribeEC2InstanceLimits](#) – 擷取目前 AWS 帳戶和目前用量層級允許的執行個體數目上限。
- [DescribeFleetCapacity](#) – 擷取機群主要區域的目前容量設定。
- [DescribeFleetLocationCapacity](#) – 擷取多位置機群每個位置的目前容量設定。
- [UpdateFleetCapacity](#) – 手動調整機群的容量設定。
- 設定：
 - [PutScalingPolicy](#) – 開啟目標型自動調整規模或建立自訂自動調整規模政策，或更新現有政策。
 - [DescribeScalingPolicies](#) – 擷取現有的自動擴展政策。
 - [DeleteScalingPolicy](#) – 刪除自動調整規模政策，並防止其影響機群的容量。
 - [StartFleetActions](#) – 重新啟動機群的自動擴展政策。
 - [StopFleetActions](#) – 暫停機群的自動調整規模政策。

監控機群活動。

- [DescribeFleetUtilization](#) – 擷取目前在機群上作用中之伺服器程序、遊戲工作階段和玩家數量的統計資料。
- [DescribeFleetLocationUtilization](#) – 擷取多位置機群中每個位置的使用率統計資料。

- [DescribeFleetEvents](#) – 檢視指定時間範圍內機群的記錄事件。
- [DescribeGameSessions](#) – 擷取遊戲工作階段中繼資料，包括遊戲的執行時間和目前的玩家計數。

設定遊戲工作階段放置的佇列

您可以設定多機群、多區域佇列，以使用成本、延遲性和彈性最佳的可用託管資源來放置遊戲工作階段。

- [CreateGameSessionQueue](#) – 建立佇列，以便在處理遊戲工作階段置放的請求時使用。
- [DescribeGameSessionQueues](#) – 擷取Amazon GameLift Servers區域中定義的遊戲工作階段佇列。
- [UpdateGameSessionQueue](#) – 變更遊戲工作階段佇列的組態。
- [DeleteGameSessionQueue](#) – 從區域移除遊戲工作階段佇列。

管理別名

您可以使用別名來表示機群，或是建立終端機替代目的地。當您將遊戲活動從一個機群轉移到另一個機群 (例如遊戲伺服器建置更新期間) 時，別名便可派上用場。

- [CreateAlias](#) – 定義新的別名，並選擇性地將其指派給機群。
- [ListAliases](#) – 取得Amazon GameLift Servers區域中定義的所有機群別名。
- [DescribeAlias](#) – 擷取現有別名的資訊。
- [UpdateAlias](#) – 變更別名的設定，例如將其從一個機群重新導向到另一個機群。
- [DeleteAlias](#) – 從區域移除別名。
- [ResolveAlias](#) – 取得指定別名指向的機群 ID。

連線至受管託管執行個體

您可以檢視機群中個別執行個體的資訊，或是請求遠端存取指定的機群執行個體以進行故障診斷。

- [DescribeInstances](#) – 取得機群中每個執行個體的資訊，包括執行個體 ID、IP 地址、位置和狀態。
- [GetInstanceAccess](#) – 請求從遠端連線至機群中指定執行個體所需的存取憑證。

設定 VPC 對等互連

建立和管理Amazon GameLift Servers託管資源和其他 AWS 資源之間的 VPC 互連連線。

- [CreateVpcPeeringAuthorization](#) – 授權與其中一個 VPCs 的對等連線。
- [DescribeVpcPeeringAuthorizations](#) – 擷取有效的互連連線授權。
- [DeleteVpcPeeringAuthorization](#) – 刪除對等連線授權。
- [CreateVpcPeeringConnection](#) – 在Amazon GameLift Servers機群的 VPC 與其中一個 VPCs之間建立對等連線。
- [DescribeVpcPeeringConnections](#) – 擷取與Amazon GameLift Servers機群之作用中或待定 VPC 互連連線的相關資訊。
- [DeleteVpcPeeringConnection](#) – 刪除與Amazon GameLift Servers機群的 VPC 對等互連。

開始遊戲工作階段並加入玩家

從後端服務呼叫這些操作，以啟動新的遊戲工作階段、取得現有遊戲工作階段的相關資訊，並將玩家加入遊戲工作階段。這些操作適用於 上託管的自訂遊戲伺服器Amazon GameLift Servers。如果您使用的是 Amazon GameLift ServersRealtime，請使用 管理遊戲工作階段[Amazon GameLift ServersRealtime 用戶端 API \(C#\) 參考](#)。

- 為一或多個玩家啟動新的遊戲工作階段。
 - [StartGameSessionPlacement](#) – 要求 Amazon GameLift Servers 尋找最佳的可用託管資源，並開始新的遊戲工作階段。這是建立新遊戲工作階段的偏好方法。它依賴遊戲工作階段佇列來追蹤跨多個區域的託管可用性，並使用FleetIQ演算法根據玩家延遲、託管成本、位置等來排定置放的優先順序。
 - [DescribeGameSessionPlacement](#) – 取得置放請求的詳細資訊和狀態。
 - [StopGameSessionPlacement](#) – 取消置放請求。
 - [CreateGameSession](#) – 在特定機群位置啟動新的空遊戲工作階段。此操作可讓您更好地控制從何處開始遊戲工作階段，而不是使用 FleetIQ 評估置放選項。您必須在不同的步驟中將玩家新增至新的遊戲工作階段。
- 讓玩家進入現有的遊戲工作階段。尋找具有可用玩家位置的執行中遊戲工作階段，並將其保留給新玩家。
 - [CreatePlayerSession](#) – 為玩家預留開放位置以加入遊戲工作階段。
 - [CreatePlayerSessions](#) – 為多個玩家預留開放位置，以加入遊戲工作階段。
- 使用遊戲工作階段和玩家工作階段資料。管理遊戲工作階段和玩家工作階段的相關資訊。
 - [SearchGameSessions](#) – 根據一組搜尋條件，請求作用中遊戲工作階段的清單。
 - [DescribeGameSessions](#) – 擷取特定遊戲工作階段的中繼資料，包括作用中的時間長度和目前的玩家計數。

- [DescribeGameSessionDetails](#) – 擷取一或多個遊戲工作階段的中繼資料，包括遊戲工作階段保護設定。
- [DescribePlayerSessions](#) – 取得玩家活動的詳細資訊，包括狀態、播放時間和玩家資料。
- [UpdateGameSession](#) – 變更遊戲工作階段設定，例如玩家計數上限和聯結政策。
- [GetGameSessionLogUrl](#) – 取得遊戲工作階段儲存日誌的位置。

適用於 的伺服器 SDK 5.x Amazon GameLift Servers

本節提供適用於 的伺服器 SDK 5.x 參考文件。Amazon GameLift Servers 伺服器開發套件提供遊戲伺服器用來與服務互動的核心功能 Amazon GameLift Servers。例如，您的遊戲伺服器會收到來自 服務的提示，以啟動和停止遊戲工作階段，並定期為服務提供遊戲工作階段狀態更新。將您的遊戲伺服器與伺服器 SDK 整合，再部署它們進行託管。

使用此伺服器 SDK 參考整合您的自訂多玩家遊戲伺服器，以便與 進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

適用於 的伺服器 SDK 的最新主要版本 Amazon GameLift Servers 為 5.x。下列託管功能需要使用 5.x 版：

- Amazon GameLift Servers 隨處
- Amazon GameLift Servers Unreal Engine 和 Unity 的外掛程式

主題

- [適用於 的伺服器 SDK 5 中的更新 Amazon GameLift Servers](#)
- [遷移至適用於 的伺服器 SDK 5.x Amazon GameLift Servers](#)
- [適用於 -- 動作的 C++ 伺服器 SDK Amazon GameLift Servers 5.x](#)
- [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)
- [適用於 Amazon GameLift Servers -- 動作的 Go 伺服器 SDK](#)
- [適用於 -- 動作的 C++ \(非真實\) 伺服器 SDK Amazon GameLift Servers 5.x](#)

適用於 的伺服器 SDK 5 中的更新 Amazon GameLift Servers

您的託管遊戲伺服器使用適用於 的伺服器 SDK Amazon GameLift Servers 與 Amazon GameLift Servers 服務通訊，以啟動和管理玩家的遊戲工作階段。最新版本 Amazon GameLift Servers 的伺服器

SDK 5 為新Amazon GameLift Servers功能提供了許多改善和支援。如果您的遊戲伺服器建置目前使用 Amazon GameLift Servers伺服器 SDK 4 或更早版本，請遵循本主題中的指示來更新您的遊戲。

Amazon GameLift Servers 伺服器開發套件 5.0.0 版及更高版本包含下列更新：

- 擴展語言 – 程式庫提供下列語言：C++、C#、Go。您可以建置與 Unreal Engine 搭配使用的 C++ 程式庫。
- 遊戲引擎外掛程式支援 – Unreal Engine 和 Unity Amazon GameLift Servers的獨立外掛程式需要Amazon GameLift Servers伺服器 SDK 5 程式庫。這些外掛程式提供引導式工作流程，將您的遊戲整合、測試和部署到 Amazon GameLift Servers以進行託管。請參閱 [Amazon GameLift ServersUnity 的外掛程式 \(伺服器 SDK 5.x\)](#)和 [Amazon GameLift ServersUnreal Engine 的外掛程式](#) 文件。
- Amazon GameLift ServersAnywhere 支援 – 透過 Anywhere 機群，您可以設定自己的託管資源來使用 Amazon GameLift Servers 功能（包括配對）。新增 Amazon GameLift Servers 代理程式以自動化遊戲工作階段生命週期管理。使用 Anywhere 機群搭配現場部署硬體進行生產託管，或設定測試環境以進行快速反覆的遊戲開發。請參閱 [隨處託管](#)和 [Amazon GameLift Servers 代理](#)程式。
- 更新的測試工具 – Amazon GameLift ServersAnywhere 功能可讓您為遊戲設定本機或雲端測試環境。設定使用或不使用 Amazon GameLift Servers 代理程式的測試。這些工具會取代 Amazon GameLift Servers Local。請參閱 [使用 Amazon GameLift ServersAnywhere 設定本機測試](#)。
- 適用於 C# 的合併 .NET 解決方案 – C# 伺服器 SDK 5.1+ 在單一解決方案中支援 .NET Framework 4.6.2（從 4.6.1 升級）和 .NET 6.0。Unity 建置的程式庫提供 .NET Standard 2.1。
- 新Compute資源 – 此新資源結合了不同類型的託管資源。它包含雲端型託管資源（受管 EC2 或容器機群）和客戶控制的託管資源（Anywhere 機群）。它包含下列更新：
 - Compute 資源的新 API 呼叫包括：[ListCompute\(\)](#)、[DescribeCompute\(\)](#) 和 [GetComputeAccess\(\)](#)。這些動作會傳回任何類型的Amazon GameLift Servers機群的託管資源資訊。一般而言，對於具有使用伺服器 SDK 5.x 之遊戲伺服器的機群，請使用運算特定動作來取代執行個體特定動作。此外，這些動作適用於沒有 Amazon GameLift Servers Agent：[RegisterCompute\(\)](#)、[DeregisterCompute\(\)](#) 和 [GetComputeAuthToken\(\)](#) 的 Anywhere 機群。
 - ActiveCompute 具有 CloudWatch 維度 FleetId、Location和的新指標ComputeType。此指標會取代先前的指標 ActiveInstances。
- Amazon EC2 Systems Manager (SSM) 用於遠端存取 – 為了提高安全性，請在連線至Amazon GameLift Servers受管機群中的執行個體時，使用 SSM 而非 SSH。請參閱[遠端連線至Amazon GameLift Servers機群執行個體](#)。

遷移至適用於 的伺服器 SDK 5.x Amazon GameLift Servers

若要更新遊戲專案以使用伺服器 SDK 5.x 版，請進行下列變更：

- 為您的開發環境 [【下載網站】](#) 取得適用於 Amazon GameLift Servers 套件的最新伺服器 SDK。針對下載的套件和版本，請遵循 Readme 檔案中的安裝指示。請參閱這些搭配遊戲專案使用伺服器 SDKs 的說明。
 - [對於使用 C++、C# 或 Go 的開發環境](#)
 - [適用於 Unreal Engine 專案（僅適用於 Unreal 程式庫的 C++ 伺服器 SDK）](#)
 - [適用於 Unity 專案（僅適用於 Unity 程式庫的 C# 伺服器 SDK）](#)
 - [與 Unreal Engine 的 Amazon GameLift Servers 外掛程式搭配使用](#)
 - [與 Unity 的 Amazon GameLift Servers 外掛程式搭配使用](#)
- 更新您的伺服器程式碼，如下所示：
 - 將伺服器程式碼回呼函數 `onCreateGameSession()` 變更為 `onStartGameSession()`。
 - 視需要更新 `InitSDK()` 輸入：
 - 如果您計劃使用 Amazon GameLift Servers 代理程式將遊戲伺服器組建部署到 Amazon GameLift Servers 受管 EC2 機群或 Anywhere 機群：

呼叫 `InitSDK()` 無參數 ([C++](#)) ([C#](#)) ([Unreal](#))。此呼叫會設定運算環境以及與 Amazon GameLift Servers 服務的 WebSocket 連線。
 - 如果您打算在沒有 Amazon GameLift Servers 代理程式的情況下，將遊戲伺服器組建部署到 Anywhere 機群：

`InitSDK()` 使用伺服器參數 ([C++](#)) ([C#](#)) ([Unreal](#)) 呼叫。遊戲伺服器程序使用這些參數來建立與服務的連線 Amazon GameLift Servers。
- 如果您的遊戲伺服器建置或其他託管應用程式在執行時與其他 AWS 資源通訊，您將需要變更應用程式存取這些資源的方式。以新的伺服器 SDK `AssumeRoleCredentials` 動作 `GetFleetRoleCredentials()`（適用於遊戲伺服器）取代 `AssumeRoleCredentials` 的使用，或使用共用的登入資料（適用於其他應用程式）。如需如何實作此變更的詳細資訊，請參閱 [與機群中的其他 AWS 資源通訊](#)。
- 如果您的專案呼叫伺服器 SDK 動作 `GetInstanceCertificate()` 來擷取 TLS 憑證，請修改您的程式碼以改用新的 `GetComputeCertificate()` ([C++](#)) ([C#](#)) ([Unreal](#))。
- 將遊戲組建上傳至 Amazon GameLift Servers（例如使用 [upload-build](#) 或 [CreateBuild\(\)](#)）時，請將 `ServerSdkVersion` 參數設定為您正在使用的 5.x 版本（此參數目前預設為 4.0.2）。此參數必須

符合遊戲伺服器建置中的實際伺服器 SDK 程式庫。如果您為上傳的遊戲伺服器組建指定了錯誤版本，則使用該組建建立的任何機群都將失敗。請參閱 [部署用於Amazon GameLift Servers託管的自訂伺服器建置](#)。

下列範例說明如何指定伺服器 SDK 版本：

```
aws gamelift upload-build \  
  --operating-system AMAZON_LINUX_2023 \  
  --server-sdk-version "5.0.0" \  
  --build-root "~/mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-west-2
```

6. 如果您使用指令碼遠端連線至受管機群，請更新指令碼以使用新程序，如中所述[遠端連線至Amazon GameLift Servers機群執行個體](#)。

適用於 -- 動作的 C++ 伺服器 SDK Amazon GameLift Servers 5.x

使用伺服器 SDK 5.x 參考整合多玩家遊戲，以便與 進行託管。Amazon GameLift Servers如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

Note

本主題說明使用 Amazon GameLift Servers C++ 標準程式庫 () 建置時可以使用的 C++ APIstd。具體而言，本文件適用於您使用 -DDGAMELIFT_USE_STD=1選項編譯的程式碼。

適用於 -- 資料類型的 C++ 伺服器 SDK Amazon GameLift Servers 5.x

使用 Amazon GameLift Servers C++ 伺服器 SDK 5.x 參考來整合多玩家遊戲，以便與 進行託管。Amazon GameLift Servers如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

Note

本主題說明使用 Amazon GameLift Servers C++ 標準程式庫 () 建置時可以使用的 C++ APIstd。具體而言，本文件適用於您使用 -DDGAMELIFT_USE_STD=1選項編譯的程式碼。

適用於 -- 動作的 C++ 伺服器 SDK Amazon GameLift Servers 5.x

資料類型

- [LogParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [AttributeValue](#)
- [GetFleetRoleCredentialsRequest](#)
- [AwsLongOutcome](#)
- [AwsStringOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [GenericOutcome](#)
- [GenericOutcomeCallable](#)
- [PlayerSession](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [InitSDKOutcome](#)
- [GameLiftError](#)
- [列舉](#)

LogParameters

識別在遊戲工作階段期間產生的檔案的物件，Amazon GameLift Servers 您想要在遊戲工作階段結束後上傳和存放。遊戲伺服器會將 Amazon GameLift Servers 做為 [ProcessReady\(\)](#) 呼叫中 `ProcessParameters` 物件的一部分 `LogParameters` 提供給。

屬性	Description
LogPaths	Amazon GameLift Servers 您要存放以供日後存取的遊戲伺服器日誌檔案的目錄路徑清單。伺服器程序會在每個遊戲工作階段期間產生這些檔案。您可以在遊戲伺服器中定義檔案路徑和名稱，並將其存放在根遊戲建置目錄中。 日誌路徑必須是絕對的。例如，如果您的遊戲組建將遊戲工作階段日誌儲存在類似的路徑中 <code>MyGame\sessionLogs\</code>，則路徑會位於 Windows 執行個體 <code>c:\game\MyGame\sessionLogs</code> 上。 Type (類型) : <code>std::vector<std::string></code> 必要 : 否

ProcessParameters

此資料類型包含傳送至 Amazon GameLift Servers 的一組參數 [ProcessReady\(\)](#)。

屬性	Description
LogParameters	具有在遊戲工作階段期間產生之檔案目錄路徑的物件。會 Amazon GameLift Servers 複製和存放檔案以供日後存取。 Type (類型) : <code>Aws::GameLift::Server::</code> LogParameters 必要 : 否

OnHealthCheck

Amazon GameLift Servers 呼叫 從伺服器程序請求運作狀態報告的回呼函數。每 60 秒會 Amazon GameLift Servers 呼叫此函數，並等待 60 秒進行回應。FALSE 如果運作狀態 TRUE 不良，伺服器程序會傳回。如果未傳回任何回應，會將伺服器程序 Amazon GameLift Servers 記錄為運作狀態不佳。

Type (類型) : `std::function<bool()>`
`onHealthCheck`

必要：否

OnProcessTerminate

Amazon GameLift Servers 呼叫 以強制伺服器程序關閉的回呼函數。呼叫此函數後，會 Amazon GameLift Servers 等待 5 分鐘讓伺服器程序關閉，並在關閉伺服器程序之前回應 [ProcessEnding\(\)](#) 呼叫。

Type (類型) : `std::function<void()>`
`onProcessTerminate`

必要：是

OnStartGameSession

Amazon GameLift Servers 叫用以啟用新遊戲工作階段的回呼函數。會 Amazon GameLift Servers 呼叫此函數以回應用戶端請求 [CreateGameSession](#)。回呼函數會傳遞 [GameSession](#) 物件。

Type (類型) : `const std::function<void (Aws::GameLift::Model::GameSession)>` `onStartGameSession`

必要：是

OnUpdateGameSession	Amazon GameLift Servers 呼叫 以將更新後的遊戲工作階段物件傳遞至伺服器程序的回呼函數。會在處理配對回填請求時Amazon GameLift Servers呼叫此函數，以提供更新後的配對建構器資料。它會傳遞GameSession物件、狀態更新 (updateReason) 和相符回填票證 ID。 Type (類型) : std::function<void (Aws::GameLift::Server::Model::UpdateGameSession)> onUpdateGameSession 必要：否
連線埠	伺服器程序接聽新玩家連線的連接埠號碼。值必須屬於為部署此遊戲伺服器組建之機群所設定的連接埠範圍。此連接埠號碼包含在遊戲工作階段和遊戲工作階段物件中，遊戲工作階段會使用該物件來連接到伺服器程序。 Type (類型) : Integer 必要：是

UpdateGameSession

此資料類型會更新至遊戲工作階段物件，其中包括遊戲工作階段已更新的原因，以及如果回填用於填入遊戲工作階段中的玩家工作階段，則相關的回填票證 ID。

屬性	Description
GameSession	GameSession 物件。GameSession 物件包含描述遊戲工作階段的屬性。 Type (類型) : Aws::GameLift::Server::GameSession 必要：是

屬性	Description
UpdateReason	正在更新遊戲工作階段的原因。 Type (類型) : <code>Aws::GameLift::Server::UpdateReason</code> 必要 : 是
BackfillTicketId	嘗試更新遊戲工作階段的回填票證 ID。 Type (類型) : <code>std::string</code> 必要 : 否

GameSession

此資料類型提供遊戲工作階段的詳細資訊。

屬性	Description
GameSessionId	遊戲工作階段的唯一識別符。遊戲工作階段 ARN 的格式如下 : <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code>。 Type (類型) : <code>std::string</code> 必要 : 否
名稱	遊戲工作階段的描述性標籤。 Type (類型) : <code>std::string</code> 必要 : 否
FleetId	遊戲工作階段執行所在的機群的唯一識別符。 Type (類型) : <code>std::string</code>

屬性	Description
	必要：否
MaximumPlayerSessionCount	遊戲工作階段的玩家連線數目上限。 Type (類型) : int 必要：否
連線埠	遊戲工作階段的連接埠號碼。若要連線到 Amazon GameLift Servers 遊戲伺服器，應用程式需要 IP 地址和連接埠號碼。 Type (類型) : int 必要：否
IpAddress	遊戲工作階段的 IP 地址。若要連線到 Amazon GameLift Servers 遊戲伺服器，應用程式需要 IP 地址和連接埠號碼。 Type (類型) : std::string 必要：否
GameSessionData	一組自訂遊戲工作階段屬性，格式為單一字串值。 Type (類型) : std::string 必要：否
MatchmakerData	用於建立遊戲工作階段的配對程序資訊，以 JSON 語法格式為字串。除了使用的配對組態之外，還包含指派給配對的所有玩家的資料，包括玩家屬性和團隊指派。 Type (類型) : std::string 必要：否

屬性	Description
GameProperties	一組遊戲工作階段的自訂屬性，格式為 key : value 對。這些屬性會與啟動新遊戲工作階段的請求一起傳遞。 Type (類型) : <code>std :: vector < GameProperty ></code> 必要 : 否
DnsName	指派給執行遊戲工作階段之執行個體的 DNS 識別符。值的格式如下： - 啟用 TLS 的機群 : <code><unique identifier>.<region identifier>.amazon gamelift.com</code> 。 - Non-TLS-enabled 的機群 : <code>ec2-<unique identifier>.compute.amazonaws.com</code> 。 連線至在啟用 TLS 的機群上執行的遊戲工作階段時，您必須使用 DNS 名稱，而非 IP 地址。 Type (類型) : <code>std::string</code> 必要 : 否

ServerParameters

遊戲伺服器程序用來建立與服務連線的資訊Amazon GameLift Servers。[InitSDK\(\)](#) 只有在遊戲伺服器建置將部署到沒有 Amazon GameLift Servers代理程式的 Anywhere 機群或容器機群時，才呼叫時包含這些參數。對於所有其他部署案例，請呼叫 [InitSDK\(\)](#) 而不使用參數。

屬性	Description
webSocketUrl	當您 RegisterCompute 使用 Amazon GameLift ServersAnywhere 運算資源時，

屬性	Description
	會GameLiftServerSdkEndpoint Amazon GameLift Servers傳回。 Type (類型) : std::string 必要 : 是
processId	向託管遊戲的伺服器程序註冊的唯一識別符。 Type (類型) : std::string 必要 : 是
hostId	HostID 是當您註冊運算ComputeName 時使用的。如需詳細資訊，請參閱 RegisterCompute。 Type (類型) : std::string 必要 : 是
fleetId	運算註冊目的地機群的唯一識別符。如需詳細資訊，請參閱 RegisterCompute。 Type (類型) : std::string 必要 : 是
authToken	產生的身分驗證字符Amazon GameLift Servers，用於向 驗證您的伺服器Amazon GameLift Servers。如需詳細資訊，請參閱 GetComputeAuthToken。 Type (類型) : std::string 必要 : 是

StartMatchBackfillRequest

用來建立配對回填請求的資訊。遊戲伺服器會在[StartMatchBackfill\(\)](#)通話Amazon GameLift Servers中將此資訊傳達給。

屬性	Description
GameSessionArn	唯一的遊戲工作階段識別符。API 操作會以 ARN 格式GetGameSessionId 傳回識別符。 Type (類型) : std::string 必要 : 是
MatchmakingConfigurationArn	配對建構器用於此請求的唯一識別符，格式為 ARN。原始遊戲工作階段的配對建構器 ARN 位於配對建構器資料屬性中的遊戲工作階段物件中。進一步了解使用配對建構器資料中的配對建構器資料。 Type (類型) : std::string 必要 : 是
Players	一組資料，代表遊戲工作階段中的所有玩家。配對建構器使用此項資訊搜尋適合配對現有玩家的新玩家。 Type (類型) : std::vector<Player> 必要 : 是
TicketId	配對或配對回填請求票證的唯一識別符。如果您不提供值，Amazon GameLift Servers會產生值。您可使用此識別項依據需求追蹤配對回填票證狀態或取消要求。 Type (類型) : std::string 必要 : 否

Player

此資料類型代表配對中的玩家。啟動配對請求時，玩家會有玩家 ID、屬性和可能的延遲資料。會在配對完成後Amazon GameLift Servers新增隊伍資訊。

屬性	Description
LatencyInMS	一組以毫秒為單位的值，指出玩家連線到位置時所經歷的延遲量。 如果使用此屬性，則播放器只會符合列出的位置。若配對構建器有評估玩家延遲的規則，玩家則必須回報延遲度，方可配對。 Type (類型) : Dictionary<string,int> 必要：否
PlayerAttributes	包含玩家資訊的金鑰：值對集合，可用於配對。玩家屬性金鑰必須符合配對規則集中使用的 PlayerAttributes。 如需玩家屬性的詳細資訊，請參閱 Attribute Value。 Type (類型) : std::map<std::string,AttributeValue> 必要：否
PlayerId	玩家的唯一識別符。 Type (類型) : std::string 必要：否
團隊	玩家在配對中獲指派的隊伍名稱。您可以在配對規則集中定義團隊名稱。 Type (類型) : std::string

屬性	Description
	必要：否

DescribePlayerSessionsRequest

指定要擷取哪些玩家工作階段的物件。伺服器程序會在[DescribePlayerSessions\(\)](#)呼叫時提供此資訊 Amazon GameLift Servers。

屬性	Description
GameSessionId	唯一的遊戲工作階段識別符。請使用此參數要求特定遊戲工作階段的所有玩家工作階段。 遊戲工作階段 ID 格式為 <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code>。GameSessionID 是自訂 ID 字串或 Type (類型) : <code>std::string</code> 必要：否
PlayerSessionId	玩家工作階段的唯一識別符。使用此參數請求單一特定玩家工作階段。 Type (類型) : <code>std::string</code> 必要：否
PlayerId	玩家的唯一識別符。使用此參數來請求特定玩家的所有玩家工作階段。請參閱 產生玩家 IDs。 Type (類型) : <code>std::string</code> 必要：否
PlayerSessionStatusFilter	要篩選結果的玩家工作階段狀態。可能的玩家工作階段狀態包括：

屬性	Description
	- RESERVED – 已收到玩家工作階段請求，但玩家尚未連線到伺服器程序或已經過驗證。 - ACTIVE – 播放器已由伺服器程序驗證並連線。 - 已完成 – 玩家連線中斷。 - TIMEDOUT – 已收到玩家工作階段請求，但玩家未連線或未在逾時限制 (60 秒) 內驗證。 Type (類型) : <code>std::string</code> 必要 : 否
NextToken	表示結果下一頁開始的字符。若要指定結果集的開頭，請不要提供值。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : <code>std::string</code> 必要 : 否
限制	回傳結果的數量上限。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : <code>int</code> 必要 : 否

StopMatchBackfillRequest

用來取消配對回填請求的資訊。遊戲伺服器會在[StopMatchBackfill\(\)](#)通話中將此資訊傳達給 Amazon GameLift Servers 服務。

屬性	Description
GameSessionArn	取消請求的唯一遊戲工作階段識別符。

屬性	Description
	Type (類型) : char[] 必要 : 否
MatchmakingConfigurationArn	此請求傳送至之配對建構器的唯一識別符。 Type (類型) : char[] 必要 : 否
TicketId	要取消之回填請求票證的唯一識別符。 Type (類型) : char[] 必要 : 否

AttributeValue

在[Player](#)屬性鍵值對中使用這些值。此物件可讓您使用任何有效的資料類型來指定屬性值：字串、數字、字串陣列或資料映射。每個AttributeValue物件必須完全使用其中一個可用的屬性：S、SL、N或SDM。

屬性	描述
AttrType	指定屬性值的類型。可能的屬性值類型包括： - NONE - STRING - DOUBLE - STRING_LIST - STRING_DOUBLE_MAP 必要 : 否
S	代表字串屬性值。 Type (類型) : std::string

屬性	描述
	必要：否
N	代表數值屬性值。 Type (類型) : double 必要：否
SL	代表字串屬性值的陣列。 Type (類型) : std::vector<std::string> 必要：否
SDM	代表字串索引鍵和雙值的字典。 Type (類型) : std::map<std::string, double> 必要：否

GetFleetRoleCredentialsRequest

此資料類型可讓遊戲伺服器有限地存取您的其他 AWS 資源。如需詳細資訊，請參閱 [設定的 IAM 服務角色 Amazon GameLift Servers](#)。

屬性	Description
RoleArn	服務角色的 Amazon Resource Name (ARN) ， 可延伸 AWS 資源的有限存取權。 Type (類型) : std::string 必要：否
RoleSessionName	角色工作階段名稱，可用來唯一識別 AWS Security Token Service AssumeRole 工作階段。此名稱會在稽核日誌中公開，例如 CloudTrail 中的稽核日誌。

屬性	Description
	Type (類型) : <code>std::string</code>
	必要 : 否

AwsLongOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : <code>long</code> 必要 : 否
ResultWithOwnership	動作的結果會轉換為 <code>rvalue</code> ，以便呼叫程式碼可以取得物件的擁有權。 Type (類型) : <code>long&&</code> 必要 : 否
成功	動作是否成功。 Type (類型) : <code>bool</code> 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

AwsStringOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : <code>std::string</code> 必要 : 否
ResultWithOwnership	動作的結果會轉換為 <code>rvalue</code>，以便呼叫程式碼可以取得物件的擁有權。 Type (類型) : <code>long&&</code> 必要 : 否
成功	動作是否成功。 Type (類型) : <code>bool</code> 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

DescribePlayerSessionsOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “DescribePlayerSessionsResult” 必要 : 否
ResultWithOwnership	動作的結果會轉換為 <code>rvalue</code>，以便呼叫程式碼可以取得物件的擁有權。

屬性	描述
	Type (類型) : <code>Aws::GameLift::Server::Model::DescribePlayerSessionsResult</code> 必要 : 否
成功	動作是否成功。 Type (類型) : <code>bool</code> 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

DescribePlayerSessionsResult

包含符合請求之每個玩家工作階段屬性的物件集合。

屬性	描述
NextToken	表示結果下一個序列頁面開始的字符。使用先前呼叫此操作時傳回的字符。若要從結果集的開頭開始，請勿指定值。若指定玩家工作階段 ID，此參數將遭到忽略。 Type (類型) : <code>std::string</code> 必要 : 是
PlayerSessions	Type (類型) : <code>IList<the section called “PlayerSession”></code> 必要 :
ResultWithOwnership	動作的結果會轉換為 <code>rvalue</code> ，以便呼叫程式碼可以取得物件的擁有權。

屬性	描述
	Type (類型) : std::string&& 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

GenericOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

GenericOutcomeCallable

此資料類型是非同步的一般結果。它具有下列屬性：

屬性	描述
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

PlayerSession

此資料類型代表Amazon GameLift Servers傳遞至遊戲伺服器的玩家工作階段。如需詳細資訊，請參閱[PlayerSession](#)。

屬性	描述
CreationTime	Type (類型) : long 必要 : 否
FleetId	Type (類型) : std::string 必要 : 否
GameSessionId	Type (類型) : std::string 必要 : 否
IpAddress	Type (類型) : std::string 必要 : 否
PlayerData	Type (類型) : std::string 必要 : 否

屬性	描述
PlayerId	Type (類型) : std::string 必要 : 否
PlayerSessionId	Type (類型) : std::string 必要 : 否
連線埠	Type (類型) : int 必要 : 否
Status	用於篩選結果的玩家工作階段狀態。提供 PlayerSessionId 或 PlayerId 時，則 PlayerSessionStatusFilter 不會影響回應。 類型 : PlayerSessionStatus 列舉。可能的值包括以下 : • ACTIVE • COMPLETED (已完成) • NOT_SET • RESERVED • 逾時 必要 : 否
TerminationTime	Type (類型) : long 必要 : 否
DnsName	Type (類型) : std::string 必要 : 否

StartMatchBackfillOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “StartMatchBackfillResult” 必要 : 否
ResultWithOwnership	動作的結果會轉換為 rvalue，以便呼叫程式碼可以取得物件的擁有權。 Type (類型) : StartMatchBackfillResult&& 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

StartMatchBackfillResult

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
TicketId	配對票證的唯一識別符。如果此處未指定票證 ID，Amazon GameLift Servers 將以 UUID 的形式產生票證 ID。使用此識別符來追蹤配對回填票證狀態，並擷取配對結果。 Type (類型) : std::string 必要 : 否

GetComputeCertificateOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “GetComputeCertificateResult” 必要：否
ResultWithOwnership	動作的結果會轉換為 rvalue，以便呼叫程式碼可以取得物件的擁有權。 Type (類型) : <code>Aws::GameLift::Server::Model::GetComputeCertificateResult&&</code> 必要：否
成功	動作是否成功。 Type (類型) : <code>bool</code> 必要：是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要：否

GetComputeCertificateResult

運算上 TLS 憑證的路徑和運算的主機名稱。

屬性	描述
CertificatePath	運算資源上 TLS 憑證的路徑。使用 Amazon GameLift Servers 受管機群時，此路徑包含：

屬性	描述
	• <code>certificate.pem</code> : 最終使用者憑證。完整的憑證鏈是<code>certificateChain.pem</code> 附加到此憑證的 組合。 • <code>certificateChain.pem</code> : 包含根憑證和中繼憑證的憑證鏈。 • <code>rootCertificate.pem</code> : 根憑證。 • <code>privateKey.pem</code> : 最終使用者憑證的私有金鑰。 Type (類型) : <code>std::string</code> 必要 : 否
ComputeName	運算資源的名稱。 Type (類型) : <code>std::string</code> 必要 : 否

GetFleetRoleCredentialsOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “GetFleetRoleCredentialsResult” 必要 : 否
ResultWithOwnership	動作的結果會轉換為 <code>rvalue</code>，以便呼叫程式碼可以取得物件的擁有權。 Type (類型) : <code>Aws::GameLift::Server::Model::GetFleetRoleCredentialsResult</code> 必要 : 否

屬性	描述
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

GetFleetRoleCredentialsResult

屬性	描述
AccessKeyId	驗證和提供 AWS 資源存取權的存取金鑰 ID。 Type (類型) : string 必要 : 否
AssumedRoleId	服務角色所屬的使用者 ID。 Type (類型) : string 必要 : 否
AssumedRoleUserArn	服務角色所屬使用者的 Amazon Resource Name (ARN)。 Type (類型) : string 必要 : 否
過期	工作階段登入資料過期前的時間長度。 Type (類型) : DateTime 必要 : 否

屬性	描述
SecretAccessKey	用於身分驗證的私密存取金鑰 ID。 Type (類型) : string 必要 : 否
SessionToken	識別目前作用中工作階段與 AWS 資源互動的字符。 Type (類型) : string 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

InitSDKOutcome

Note

InitSDKOutcome 只有在您使用 std 旗標建置 SDK 時，才會傳回。如果您使用 nostd 旗標建置，[the section called “GenericOutcome”](#)則會改為傳回。

屬性	描述
成功	動作是否成功。 Type (類型) : bool

屬性	描述
	必要：是
錯誤	動作失敗時發生的錯誤。 Type (類型)： the section called “GameLiftError” 必要：否

GameLiftError

屬性	描述
ErrorType	錯誤類型。 類型：GameLiftErrorType 列舉 。 必要：否
ErrorMessage	錯誤的名稱。 Type (類型)：std::string 必要：否
ErrorName	錯誤訊息。 Type (類型)：std::string 必要：否

列舉

為適用於 Amazon GameLift Servers(C++) 的伺服器 SDK 定義的列舉定義如下：

GameLiftErrorType

指出錯誤類型的字串值。有效值包含：

- BAD_REQUEST_EXCEPTION

- GAMESESSION_ID_NOT_SET – 尚未設定遊戲工作階段 ID。
- INTERNAL_SERVICE_EXCEPTION
- LOCAL_CONNECTION_FAILED – 與的本機連線Amazon GameLift Servers失敗。
- NETWORK_NOT_INITIALIZED – 網路尚未初始化。
- SERVICE_CALL_FAILED – 呼叫 AWS 服務失敗。
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- ALREADY_INITIALIZED – Amazon GameLift Servers 伺服器或用戶端已使用 Initialize() 初始化。
- FLEET_MISMATCH – 目標機群與 gameSession 或 playerSession 的機群不相符。
- GAMELIFT_CLIENT_NOT_INITIALIZED – Amazon GameLift Servers用戶端尚未初始化。
- GAMELIFT_SERVER_NOT_INITIALIZED – Amazon GameLift Servers 伺服器尚未初始化。
- GAME_SESSION_ENDED_FAILED – 的 伺服器開發套件Amazon GameLift Servers無法聯絡 服務來報告遊戲工作階段已結束。
- GAME_SESSION_NOT_READY – Amazon GameLift Servers 伺服器遊戲工作階段未啟用。
- GAME_SESSION_READY_FAILED – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務來報告遊戲工作階段已就緒。
- INITIALIZATION_MISMATCH – 在 Server : : Initialize() 之後呼叫用戶端方法，反之亦然。
- NOT_INITIALIZED – Amazon GameLift Servers 伺服器或用戶端尚未使用 Initialize() 初始化。
- NO_TARGET_ALIASID_SET – 尚未設定目標aliasId。
- NO_TARGET_FLEET_SET – 尚未設定目標機群。
- PROCESS_ENDING_FAILED – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務來報告程序即將結束。
- PROCESS_NOT_ACTIVE – 伺服器程序尚未啟用、未繫結至 GameSession，且無法接受或處理 PlayerSessions。
- PROCESS_NOT_READY – 伺服器程序尚未準備好啟用。
- PROCESS_READY_FAILED – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務以報告程序已就緒。
- SDK_VERSION_DETECTION_FAILED – SDK 版本偵測失敗。

- STX_CALL_FAILED – 呼叫 XStx 伺服器後端元件失敗。
- STX_INITIALIZATION_FAILED – XStx 伺服器後端元件無法初始化。
- UNEXPECTED_PLAYER_SESSION – 伺服器遇到未註冊的玩家工作階段。
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE – 將訊息傳送至 GameLift Service WebSocket 的可重試失敗。
- WEBSOCKET_SEND_MESSAGE_FAILURE – 無法傳送訊息至 GameLift Service WebSocket。
- MATCH_BACKFILL_REQUEST_VALIDATION – 驗證請求失敗。
- PLAYER_SESSION_REQUEST_VALIDATION – 驗證請求失敗。

PlayerSessionCreationPolicy

字串值代表遊戲工作階段是否可接受新玩家。有效值包含：

- ACCEPT_ALL – 接受所有新玩家工作階段。
- DENY_ALL – 拒絕所有新玩家工作階段。
- NOT_SET – 遊戲工作階段未設定為接受或拒絕新的玩家工作階段。

[適用於 -- 資料類型的 C++ 伺服器 SDK Amazon GameLift Servers 5.x](#)

主題

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)

- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Destroy\(\)](#)

GetSdkVersion()

傳回內建至伺服器程序的目前開發套件版本編號。

語法

```
Aws::GameLift::AwsStringOutcome Server::GetSdkVersion();
```

傳回值

如果成功，將目前開發套件版本以 [the section called “AwsStringOutcome”](#) 物件傳回。傳回的物件包含版本編號（範例 5.0.0）。如果不成功，則會傳回錯誤訊息。

範例

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

初始化 Amazon GameLift Servers 開發套件。在啟動時，在與相關的任何其他初始化步驟之前呼叫此方法 Amazon GameLift Servers。此動作會從主機環境讀取伺服器參數，以設定遊戲伺服器程序與服務之間的通訊 Amazon GameLift Servers。它使用等冪符記，因此您可以在失敗時安全地重試此呼叫。

如果遊戲伺服器組建在沒有 Amazon GameLift Servers 代理程式的情況下部署到 Amazon GameLift Servers Anywhere 機群或容器機群，請呼叫 [InitSDK\(\)](#) 並指定一組伺服器參數。

語法

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK();
```

傳回值

傳回 [the section called “InitSDKOutcome”](#) 物件，指出伺服器程序是否已準備好呼叫 [ProcessReady\(\)](#)。

範例

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
    Aws::GameLift::Server::InitSDK();
```

InitSDK()

初始化 Amazon GameLift Servers 開發套件。在啟動時，在與 相關的任何其他初始化步驟之前呼叫此方法Amazon GameLift Servers。此動作需要一組伺服器參數，才能設定遊戲伺服器程序與服務之間的通訊Amazon GameLift Servers。它使用等冪符記，因此您可以在失敗時安全地重試此呼叫。

如果遊戲伺服器建置將部署到Amazon GameLift Servers受管 EC2 機群，或使用 Amazon GameLift Servers 代理程式部署到 Amazon GameLift ServersAnywhere 機群或容器機群，請呼叫 [InitSDK\(\)](#) 而不使用伺服器參數。

語法

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK(serverParameters);
```

參數

[ServerParameters](#)

若要在 Amazon GameLift ServersAnywhere 機群上初始化遊戲伺服器，請使用下列資訊建構ServerParameters物件：

- 用來連線至遊戲伺服器的 WebSocket URL。
- 用於託管遊戲伺服器的程序 ID。

- 託管遊戲伺服器程序的運算 ID。
- 包含 Amazon GameLift ServersAnywhere 運算的 Amazon GameLift Servers 機群 ID。
- Amazon GameLift Servers 操作產生的授權字符。

傳回值

傳回 [the section called “InitSDKOutcome”](#) 物件，指出伺服器程序是否已準備好呼叫 [ProcessReady\(\)](#)。

Note

如果部署到 Anywhere 機群的遊戲組建呼叫 `InitSDK()` 失敗，請檢查建立組建資源時使用的 `ServerSdkVersion` 參數。您必須將此值明確設定為正在使用的伺服器 SDK 版本。此參數的預設值為 4.x，不相容。若要解決此問題，請建立新的組建，並將其部署到新的機群。

範例

Amazon GameLift ServersAnywhere 範例

```
//Define the server parameters
std::string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
std::string processId = "PID1234";
std::string fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
std::string hostId = "HardwareAnywhere";
std::string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
Aws::GameLift::Server::Model::ServerParameters serverParameters =
    Aws::GameLift::Server::Model::ServerParameters(webSocketUrl, authToken, fleetId,
        hostId, processId);

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
    Aws::GameLift::Server::InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift Servers 通知伺服器程序已準備好託管遊戲工作階段。叫用 後呼叫此方法 [InitSDK\(\)](#)。每個程序只能呼叫此方法一次。

語法

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

參數

processParameters

[ProcessParameters](#) 物件會傳達以下有關伺服器程序的資訊：

- 在遊戲伺服器程式碼中實作的回呼方法名稱，Amazon GameLift Servers服務會叫用它來與伺服器程序通訊。
- 伺服器程序正在接聽的埠號。
- 任何要 Amazon GameLift Servers 擷取並存放的遊戲工作階段特定檔案路徑。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會說明 [ProcessReady\(\)](#) 呼叫和委派函數的實作。

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");           // Example of a log file written by the
game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
        listenPort,
        Aws::GameLift::Server::LogParameters(logPaths)
    );

Aws::GameLift::GenericOutcome outcome =
    Aws::GameLift::Server::ProcessReady(processReadyParameter);
```

```
// Implement callback functions
void Server::onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome =
        Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void Server::onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool Server::onHealthCheck()
{
    bool health;
    // complete health evaluation within 60 seconds and set health
    return health;
}
```

ProcessReadyAsync()

若伺服器程序已準備好託管遊戲工作階段，則此動作會通知 Amazon GameLift Servers 服務。此方法應在伺服器程序準備好託管遊戲工作階段後呼叫。參數會指定在特定情況下 Amazon GameLift Servers 呼叫的回呼函數名稱。遊戲伺服器代碼必須實作上述函數。

此為非同步呼叫。若要進行同步呼叫，請使用 [ProcessReady\(\)](#)。如需詳細資訊，請參閱[初始化伺服器程序](#)。

語法

```
GenericOutcomeCallable ProcessReadyAsync(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

參數

processParameters

[ProcessParameters](#) 物件會傳達以下有關伺服器程序的資訊：

- 在遊戲伺服器程式碼中實作的回呼方法名稱，Amazon GameLift Servers服務會叫用它來與伺服器程序通訊。
- 伺服器程序正在接聽的埠號。
- 任何要 Amazon GameLift Servers 擷取並存放的遊戲工作階段特定檔案路徑。

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");           // This is an example of a log file
written by the game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(std::bind(&Server::onStartGameSession, this,
std::placeholders::_1),
std::bind(&Server::onProcessTerminate, this), std::bind(&Server::OnHealthCheck,
this),
std::bind(&Server::OnUpdateGameSession, this), listenPort,
Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcomeCallable outcome =
    Aws::GameLift::Server::ProcessReadyAsync(processReadyParameter);

// Implement callback functions
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
```



```
GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool onHealthCheck()
{
    // perform health evaluation and complete within 60 seconds
    return health;
}
```

ProcessEnding()

Amazon GameLift Servers 通知伺服器程序正在終止。在所有其他清除任務（包括關閉作用中的遊戲工作階段）之後，以及在終止程序之前，呼叫此方法。根據的結果 `ProcessEnding()`，程序會以成功 (0) 或錯誤 (-1) 結束並產生機群事件。如果程序因錯誤而終止，產生的機群事件為 `SERVER_PROCESS_TERMINATED_UNHEALTHY`。

語法

```
Aws::GameLift::GenericOutcome processEndingOutcome =
    Aws::GameLift::Server::ProcessEnding();
```

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會先呼叫 `ProcessEnding()` 和 `Destroy()` 再以成功或錯誤結束程式碼終止伺服器程序。

```
Aws::GameLift::GenericOutcome processEndingOutcome =
    Aws::GameLift::Server::ProcessEnding();
Aws::GameLift::Server::Destroy();

// Exit the process with success or failure
if (processEndingOutcome.IsSuccess()) {
    exit(0);
}
else {
    cout << "ProcessEnding() failed. Error: " <<
        processEndingOutcome.GetError().GetErrorMessage();
    exit(-1);
}
```

ActivateGameSession()

Amazon GameLift Servers 通知伺服器程序已啟用遊戲工作階段，現在已準備好接收玩家連線。在所有遊戲工作階段初始化之後，應該呼叫此動作做為回onStartGameSession()呼函數的一部分。

語法

```
Aws::GameLift::GenericOutcome activateGameSessionOutcome =  
    Aws::GameLift::Server::ActivateGameSession();
```

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例顯示ActivateGameSession()稱為onStartGameSession()委派函數的一部分。

```
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)  
{  
    // game-specific tasks when starting a new game session, such as loading map  
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession();  
}
```

UpdatePlayerSessionCreationPolicy()

更新目前遊戲工作階段的能力，以接受新的玩家工作階段。遊戲工作階段可設定為接受或拒絕所有新的玩家工作階段。

語法

```
GenericOutcome  
UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCreationPolicy  
    newPlayerSessionPolicy);
```

參數

playerCreationSessionPolicy

類型：PlayerSessionCreationPolicy[列舉](#)值。

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例設定目前遊戲工作階段的加入政策為可接受所有玩家。

```
Aws::GameLift::GenericOutcome outcome =  
  
    Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

GetGameSessionId()

擷取作用中伺服器程序託管的遊戲工作階段 ID。

對於未使用遊戲工作階段啟用的閒置程序，呼叫會傳回 [the section called “GameLiftError”](#)。

語法

```
AwsStringOutcome GetGameSessionId()
```

參數

此動作沒有參數。

傳回值

如果成功，則會把遊戲工作階段 ID 當成 [the section called “AwsStringOutcome”](#) 物件傳回。如果不成功，則會傳回錯誤訊息。

對於未使用遊戲工作階段啟用的閒置程序，呼叫會傳回 Success=True 和 GameSessionId=""。

範例

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetTerminationTime()

若設有終止時間，即傳回伺服器程序排定關閉的時間。伺服器程序在收到來自的 `onProcessTerminate()` 呼後會採取動作 Amazon GameLift Servers。Amazon GameLift `ServersonProcessTerminate()` 會呼叫，原因如下：

- 當伺服器程序回報運作狀態不佳或未回應 時Amazon GameLift Servers。
- 在縮減規模事件期間終止執行個體時。
- 當執行個體因 [spot-instance 中斷](#) 而終止時。

語法

```
AwsDateTimeOutcome GetTerminationTime()
```

傳回值

如果成功，會將終止時間傳回為 `AwsDateTimeOutcome` 物件。此值為終止時間，以自 以來經過的刻度表示0001 00:00:00。例如，日期時間值2020-09-13 12:26:40 -000Z等於6373559680000000000刻度。如果沒有可用的終止時間，會傳回錯誤訊息。

如果程序尚未收到 `ProcessParameters.OnProcessTerminate()` 回呼，則會傳回錯誤訊息。如需關閉伺服器程序的詳細資訊，請參閱 [回應伺服器程序關閉通知](#)。

範例

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift Servers 通知具有指定玩家工作階段 ID 的玩家已連線至伺服器程序，且需要驗證。Amazon GameLift Servers 會驗證玩家工作階段 ID 是否有效。驗證玩家工作階段後，會將玩家位置 Amazon GameLift Servers的狀態從 RESERVED 變更為 ACTIVE。

語法

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

參數

playerSessionId

建立新玩家工作階段Amazon GameLift Servers時由 發出的唯一 ID。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會處理連線請求，其中包含驗證和拒絕非有效的玩家工作階段 IDs。

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
playerSessionId)
{
    Aws::GameLift::GenericOutcome connectOutcome =
    Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
    if(connectOutcome.IsSuccess())
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(connectOutcome.GetError().GetMessage());
    }
}
```

RemovePlayerSession()

Amazon GameLift Servers 通知玩家已中斷與伺服器程序的連線。作為回應，Amazon GameLift Servers 會將玩家位置變更為可用。

語法

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

參數

playerSessionId

建立新玩家工作階段Amazon GameLift Servers時由 發出的唯一 ID。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

擷取玩家工作階段資料，其中包括設定、工作階段中繼資料和玩家資料。使用此方法取得下列相關資訊：

- 單一玩家工作階段
- 遊戲工作階段中的所有玩家工作階段
- 與單一玩家 ID 相關聯的所有玩家工作階段

語法

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest  
    describePlayerSessionsRequest)
```

參數

[DescribePlayerSessionsRequest](#)

描述要擷取哪些玩家工作階段的 [the section called “DescribePlayerSessionsRequest”](#) 物件。

傳回值

如果成功，會傳回 [the section called “DescribePlayerSessionsOutcome”](#) 物件，內含一組與請求參數相符的玩家工作階段物件。

範例

此範例會請求主動連線到指定遊戲工作階段的所有玩家工作階段。透過省略 NextToken 並將限制值設定為 10，會 Amazon GameLift Servers 傳回符合請求的前 10 個玩家工作階段記錄。

```
// Set request parameters  
Aws::GameLift::Server::Model::DescribePlayerSessionsRequest request;  
request.SetPlayerSessionStatusFilter(Aws::GameLift::Server::Model::PlayerSessionStatusMapper::G  
request.SetLimit(10);
```

```
request.SetGameSessionId("the game session ID");    // can use GetGameSessionId()

// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::DescribePlayerSessions(request);
```

StartMatchBackfill()

此動作會傳送請求，以便替 FlexMatch 所建立的遊戲工作階段開放空位找到新玩家。如需詳細資訊，請參閱[FlexMatch 回填功能](#)。

此為非同步動作。如果新玩家相符，會使用回呼函數 Amazon GameLift Servers 交付更新的配對建構器資料 `OnUpdateGameSession()`。

一個伺服器程序一次僅能有一個使用中的配對回填請求。若要發送新請求，請先呼叫 [StopMatchBackfill\(\)](#) 取消原始請求。

語法

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
startBackfillRequest);
```

參數

[StartMatchBackfillRequest](#)

StartMatchBackfillRequest 物件，可傳達下列資訊：

- 指派給回填請求的票證 ID。此資訊是選用的；如果未提供 ID，Amazon GameLift Servers 將產生一個 ID。
- 傳送請求對象的配對建構器。必須填入完整的組態 ARN。此值位於遊戲工作階段的配對建構器資料中。
- 要回填的遊戲工作階段 ID。
- 遊戲工作階段目前玩家的可用配對資料。

傳回值

傳回具有相符回填票證 ID 或失敗並顯示錯誤訊息的 [the section called “StartMatchBackfillOutcome”](#) 物件。

範例

```
// Build a backfill request
std::vector<Player> players;
Aws::GameLift::Server::Model::StartMatchBackfillRequest startBackfillRequest;
startBackfillRequest.SetTicketId("1111aaaa-22bb-33cc-44dd-5555eeee66ff"); // optional,
    autogenerated if not provided
startBackfillRequest.SetMatchmakingConfigurationArn("arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"); //from the game
    session matchmaker data
startBackfillRequest.SetGameSessionArn("the game session ARN");           // can use
    GetGameSessionId()
startBackfillRequest.SetPlayers(players);                                  // from the
    game session matchmaker data

// Send backfill request
Aws::GameLift::StartMatchBackfillOutcome backfillOutcome =
    Aws::GameLift::Server::StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void Server::OnUpdateGameSession(Aws::GameLift::Server::Model::GameSession gameSession,
    Aws::GameLift::Server::Model::UpdateReason updateReason, std::string backfillTicketId)
{
    // handle status messages
    // perform game-specific tasks to prep for newly matched players
}
```

StopMatchBackfill()

取消作用中的配對回填請求。如需詳細資訊，請參閱[FlexMatch回填功能](#)。

語法

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

參數

[StopMatchBackfillRequest](#)

識別要取消之配對票證的 StopMatchBackfillRequest 物件：

- 指派給回填請求的票證 ID。

- 回填請求傳送到的配對建構器。
- 與回填請求相關聯的遊戲工作階段。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
// Set backfill stop request parameters

Aws::GameLift::Server::Model::StopMatchBackfillRequest stopBackfillRequest;
stopBackfillRequest.SetTicketId("1111aaaa-22bb-33cc-44dd-5555eeee66ff");
stopBackfillRequest.SetGameSessionArn("the game session ARN"); // can use
    GetGameSessionId()
stopBackfillRequest.SetMatchmakingConfigurationArn("arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig");
// from the game session matchmaker data

Aws::GameLift::GenericOutcome stopBackfillOutcome =
    Aws::GameLift::Server::StopMatchBackfill(stopBackfillRequest);
```

GetComputeCertificate()

擷取用於加密 Amazon GameLift ServersAnywhere 運算資源與 之間網路連線的 TLS 憑證路徑 Amazon GameLift Servers。您可以在向 Amazon GameLift ServersAnywhere 機群註冊運算裝置時，使用憑證路徑。如需詳細資訊，請參閱 [RegisterCompute](#)。

語法

```
GetComputeCertificateOutcome Server::GetComputeCertificate()
```

傳回值

傳回 [the section called “GetComputeCertificateOutcome”](#)。

範例

```
Aws::GameLift::GetComputeCertificateOutcome certificate =
    Aws::GameLift::Server::GetComputeCertificate();
```

GetFleetRoleCredentials()

擷取授權 與其他 互動的 IAM 角色登入Amazon GameLift Servers資料 AWS 服務。如需詳細資訊，請參閱[與機群中的其他 AWS 資源通訊](#)。

語法

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest request);
```

參數

[GetFleetRoleCredentialsRequest](#)

傳回值

其會傳回 [the section called “GetFleetRoleCredentialsOutcome”](#) 物件。

範例

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
    getFleetRoleCredentialsRequest;
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
    Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

此範例顯示使用選用RoleSessionName值將名稱指派給登入資料工作階段，以供稽核之用。如果您未提供角色工作階段名稱，則會使用預設值 ***#fleet-id#-#host-id#***。

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
    getFleetRoleCredentialsRequest;
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction");
getFleetRoleCredentialsRequest.SetRoleSessionName("MyFleetRoleSession");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
    Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Destroy()

從記憶體釋放Amazon GameLift Servers遊戲伺服器 SDK。最佳實務是在終止程序之後ProcessEnding()和之前呼叫此方法。如果您使用的是 Anywhere 機群，而且不是在每次遊戲工作階段後終止伺服器程序，請呼叫 Destroy()，然後在通知 Amazon GameLift Servers 程序已準備好使用 託管遊戲工作階段之前InitSDK()重新初始化 ProcessReady()。

語法

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

參數

沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
Aws::GameLift::GenericOutcome processEndingOutcome =  
    Aws::GameLift::Server::ProcessEnding();  
Aws::GameLift::Server::Destroy();  
  
// Exit the process with success or failure  
if (processEndingOutcome.IsSuccess()) {  
    exit(0);  
}  
else {  
    cout << "ProcessEnding() failed. Error: " <<  
    processEndingOutcome.GetError().GetErrorMessage();  
    exit(-1);  
}
```

適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x

使用伺服器 SDK 5.x 參考整合多玩家遊戲，以便與 進行託管。Amazon GameLift Servers如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。如果您使用的是 Unity 的 Amazon GameLift Servers 外掛程式，另請參閱 [Amazon GameLift ServersUnity 的外掛程式 \(伺服器 SDK 5.x\)](#)。

適用於 -- 資料類型的 C# 伺服器 SDK Amazon GameLift Servers 5.x

使用 Amazon GameLift Servers C# 伺服器 SDK 5.x 參考來整合多玩家遊戲，以便與 進行託管。Amazon GameLift Servers如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。如果您使用的是 Unity 的 Amazon GameLift Servers 外掛程式，另請參閱 [Amazon GameLift ServersUnity 的外掛程式 \(伺服器 SDK 5.x\)](#)。

[適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)

資料類型

- [LogParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)
- [AttributeValue](#)
- [AwsStringOutcome](#)
- [GenericOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [PlayerSession](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)

- [GetFleetRoleCredentialsResult](#)
- [AwsDateTimeOutcome](#)
- [GameLiftError](#)
- [列舉](#)

LogParameters

使用此資料類型來識別遊戲工作階段期間產生的檔案，您希望遊戲伺服器在遊戲工作階段結束後上傳到Amazon GameLift Servers這些檔案。遊戲伺服器會在[ProcessReady\(\)](#)呼叫LogParameters to Amazon GameLift Servers中通訊。

屬性	Description
LogPaths	Amazon GameLift Servers 您要存放以供日後存取的遊戲伺服器日誌檔案的目錄路徑清單。伺服器程序會在每個遊戲工作階段期間產生這些檔案。您可以在遊戲伺服器中定義檔案路徑和名稱，並將其存放在根遊戲建置目錄中。 日誌路徑必須是絕對的。例如，如果您的遊戲組建將遊戲工作階段日誌儲存在類似的路徑中MyGame\sessionLogs\，則該路徑將位於 Windows 執行個體c:\game\MyGame\sessionLogs 上。 Type (類型) : List<String> 必要：否

ProcessParameters

此資料類型包含在[ProcessReady\(\)](#)呼叫Amazon GameLift Servers中傳送至 的一組參數。

屬性	Description
LogParameters	具有遊戲工作階段日誌檔案目錄路徑清單的物件。

	Type (類型) : <code>Aws::GameLift::Server:: LogParameters</code> 必要 : 是
OnHealthCheck	Amazon GameLift Servers 呼叫 從伺服器程序請求運作狀態報告的回呼函數名稱。每 60 秒 Amazon GameLift Servers 會呼叫此函數。呼叫此函數 Amazon GameLift Servers 等待 60 秒等待回應後，如果未收到回應，會將伺服器程序 Amazon GameLift Servers 記錄為運作狀態不佳。 Type (類型) : <code>void OnHealthCheckDelegate()</code> 必要 : 是
OnProcessTerminate	Amazon GameLift Servers 叫用以強制伺服器程序關閉的回呼函數名稱。呼叫此函數後，Amazon GameLift Servers 會等待五分鐘，讓系統能關閉伺服器程序，並透過 ProcessEnding() 呼叫進行回應，再將伺服器程序關閉。 Type (類型) : <code>void OnProcessTerminateDelegate()</code> 必要 : 是
OnStartGameSession	Amazon GameLift Servers 叫用以啟用新遊戲工作階段的回呼函數名稱。會 Amazon GameLift Servers 呼叫此函數以回應用戶端請求 CreateGameSession。回呼函數接受 GameSession 物件。 Type (類型) : <code>void OnStartGameSessionDelegate(GameSession)</code> 必要 : 是

OnUpdateGameSession	Amazon GameLift Servers 叫用以將更新後的遊戲工作階段物件傳遞至伺服器程序的回呼函數名稱。會在處理配對回填請求時Amazon GameLift Servers呼叫此函數，以提供更新的配對建構器資料。它會傳遞GameSession物件、狀態更新 (updateReason) 和相符回填票證 ID。 類型：作廢 OnUpdateGameSessionDelegate (UpdateGameSession) 必要：否
連線埠	伺服器程序接聽新玩家連線的連接埠號碼。值必須屬於為部署此遊戲伺服器組建之機群所設定的連接埠範圍。此連接埠號碼包含在遊戲工作階段和遊戲工作階段物件中，遊戲工作階段會使用該物件來連接到伺服器程序。 Type (類型)：Integer 必要：是

UpdateGameSession

更新遊戲工作階段物件的資訊，包括遊戲工作階段更新的原因。如果更新與相符回填動作相關，則此資料類型會包含回填票證 ID。

屬性	Description
GameSession	GameSession 物件。GameSession 物件包含描述遊戲工作階段的屬性。 Type (類型)：GameSession GameSession() 必要：是
UpdateReason	正在更新遊戲工作階段的原因。

屬性	Description
	Type (類型) : UpdateReason UpdateReason() 必要 : 是
BackfillTicketId	嘗試更新遊戲工作階段의 回填票證 ID。 Type (類型) : String 必要 : 是

GameSession

遊戲工作階段的詳細資訊。

屬性	Description
GameSessionId	遊戲工作階段的唯一識別符。遊戲工作階段 ARN 的格式如下 : <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> 。 Type (類型) : String 必要 : 否
名稱	遊戲工作階段的描述性標籤。 Type (類型) : String 必要 : 否
FleetId	遊戲工作階段執行所在的機群的唯一識別符。 Type (類型) : String 必要 : 否

屬性	Description
MaximumPlayerSessionCount	遊戲工作階段的玩家連線數目上限。 Type (類型) : Integer 必要 : 否
連線埠	遊戲工作階段的連接埠號碼。若要連線到 Amazon GameLift Servers 遊戲伺服器，應用程式需要 IP 地址和連接埠號碼。 Type (類型) : Integer 必要 : 否
IpAddress	遊戲工作階段的 IP 地址。若要連線到 Amazon GameLift Servers 遊戲伺服器，應用程式需要 IP 地址和連接埠號碼。 Type (類型) : String 必要 : 否
GameSessionData	一組自訂遊戲工作階段屬性，格式為單一字串值。 Type (類型) : String 必要 : 否
MatchmakerData	用於建立遊戲工作階段的配對程序資訊，以 JSON 語法格式為字串。除了使用的配對組態之外，還包含指派給配對之所有玩家的資料，包括玩家屬性和團隊指派。 Type (類型) : String 必要 : 否

屬性	Description
GameProperties	一組遊戲工作階段的自訂屬性，格式為 key : value 對。這些屬性會與啟動新遊戲工作階段的請求一起傳遞。 Type (類型) : Dictionary<string, string> 必要 : 否
DnsName	指派給執行遊戲工作階段之執行個體的 DNS 識別符。值的格式如下： - 啟用 TLS 的機群 : <unique identifier>.<region identifier>.amazon gamelift.com 。 - Non-TLS-enabled 的機群 : ec2-<unique identifier>.compute.amazonaws.com 。 連線至在啟用 TLS 的機群上執行的遊戲工作階段時，您必須使用 DNS 名稱，而非 IP 地址。 Type (類型) : String 必要 : 否

ServerParameters

用來維護 Amazon GameLift ServersAnywhere 伺服器與 Amazon GameLift Servers服務之間連線的資訊。使用 啟動新的伺服器程序時，會使用此資訊[InitSDK\(\)](#)。對於託管在 Amazon GameLift Servers 受管 EC2 執行個體上的伺服器，請使用空物件。

屬性	Description
WebSocketUrl	當您 RegisterCompute 作為 Amazon GameLift ServersAnywhere 的一部分

屬性	Description
	時GameLiftServerSdkEndpoint 傳回的。 Type (類型) : String 必要 : 是
ProcessId	向託管遊戲的伺服器程序註冊的唯一識別符。 Type (類型) : String 必要 : 是
HostId	主機的唯一識別符，其中包含託管遊戲的伺服器程序。hostId 是您註冊運算時使用的 ComputeName。如需詳細資訊，請參閱 RegisterCompute Type (類型) : String 必要 : 是
FleetId	註冊運算之機群的機群 ID。如需詳細資訊，請參閱 RegisterCompute。 Type (類型) : String 必要 : 是
AuthToken	產生的身分驗證字符Amazon GameLift Servers，用於向 驗證您的伺服器Amazon GameLift Servers。如需詳細資訊，請參閱 GetComputeAuthToken。 Type (類型) : String 必要 : 是

StartMatchBackfillRequest

用來建立配對回填請求的資訊。遊戲伺服器會在[StartMatchBackfill\(\)](#) 通話 Amazon GameLift Servers 中將此資訊傳達給。

屬性	Description
GameSessionArn	唯一的遊戲工作階段識別符。API 操作會以 ARN 格式GetGameSessionId 傳回識別符。 Type (類型) : String 必要 : 是
MatchmakingConfigurationArn	配對建構器用於此請求的唯一識別符，格式為 ARN。原始遊戲工作階段的配對建構器 ARN 位於配對建構器資料屬性中的遊戲工作階段物件中。進一步了解使用配對建構器資料中的配對建構器資料。 Type (類型) : String 必要 : 是
Players	一組資料，代表目前在遊戲工作階段中的所有玩家。配對建構器使用此項資訊搜尋適合配對現有玩家的新玩家。 Type (類型) : List<Player> 必要 : 是
TicketId	配對或配對回填請求票證的唯一識別符。如果您不提供值，Amazon GameLift Servers 會產生值。您可使用此識別項依據需求追蹤配對回填票證狀態或取消要求。 Type (類型) : String 必要 : 否

Player

代表配對中的玩家。當配對請求開始時，玩家會有玩家 ID、屬性和可能的延遲資料。會在配對完成後 Amazon GameLift Servers 新增隊伍資訊。

屬性	Description
LatencyInMS	一組以毫秒為單位的值，表示玩家連線到某個位置時所經歷的延遲量。 如果使用此屬性，則播放器只會符合列出的位置。若配對構建器有評估玩家延遲的規則，玩家則必須回報延遲度，方可配對。 Type (類型) : Dictionary<string, int> 必要：否
PlayerAttributes	包含玩家資訊的金鑰：值對集合，可用於配對。玩家屬性金鑰必須符合配對規則集中使用的 PlayerAttributes。 如需玩家屬性的詳細資訊，請參閱 Attribute Value。 Type (類型) : Dictionary<string, AttributeValue> 必要：否
PlayerId	玩家的唯一識別符。 Type (類型) : String 必要：否
團隊	玩家在配對中獲指派的隊伍名稱。您可以在配對規則集中定義團隊名稱。 Type (類型) : String

屬性	Description
	必要：否

DescribePlayerSessionsRequest

此資料類型用於指定要擷取的玩家工作階段，可透過多種方式使用：(1) 提供 `PlayerSessionId` 要求特定玩家工作階段；(2) 提供 `GameSessionId` 要求指定遊戲工作階段之中的所有玩家工作階段；或 (3) 提供 `PlayerId` 要求指定玩家的所有玩家工作階段。對於大量的玩家工作階段，可使用分頁參數擷取結果做為循序頁面。

屬性	Description
<code>GameSessionId</code>	唯一的遊戲工作階段識別符。請使用此參數要求特定遊戲工作階段的所有玩家工作階段。遊戲工作階段 ID 格式如下：<code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code>。<code><ID string></code> 的值可能是自訂 ID 字串 (若在建立遊戲工作階段時有指定 ID)，或是產生的字串。 Type (類型) : String 必要：否
<code>PlayerSessionId</code>	玩家工作階段的唯一識別符。 Type (類型) : String 必要：否
<code>PlayerId</code>	玩家的唯一識別符。請參閱 產生玩家 IDs。 Type (類型) : String 必要：否
<code>PlayerSessionStatusFilter</code>	要篩選結果的玩家工作階段狀態。可能的玩家工作階段狀態包括下列項目：

屬性	Description
	- RESERVED – 玩家工作階段要求已收到，但玩家尚未連線至伺服器程序及/或通過驗證。 - ACTIVE – 玩家已由伺服器程序驗證，目前已連線。 - COMPLETED – 玩家連線已中斷。 - TIMEDOUT – 玩家工作階段要求已收到，但玩家並未在逾時限制 (60 秒) 內連線及/或通過驗證。 Type (類型) : String 必要 : 否
NextToken	表示結果下一頁開始的字符。若要指定結果集的開頭，請不要提供值。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : String 必要 : 否
限制	回傳結果的數量上限。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : int 必要 : 否

StopMatchBackfillRequest

用來取消配對回填請求的資訊。遊戲伺服器會將此資訊傳達給[StopMatchBackfill\(\)](#)通話中的 Amazon GameLift Servers 服務。

屬性	Description
GameSessionArn	取消請求的唯一遊戲工作階段識別符。

屬性	Description
	Type (類型) : string 必要 : 是
MatchmakingConfigurationArn	傳送此請求之配對建構器的唯一識別符。 Type (類型) : string 必要 : 是
TicketId	要取消之回填請求票證的唯一識別符。 Type (類型) : string 必要 : 是

GetFleetRoleCredentialsRequest

此資料類型可讓遊戲伺服器有限地存取您的其他 AWS 資源。如需詳細資訊，請參閱 [設定的 IAM 服務角色 Amazon GameLift Servers](#)。

屬性	Description
RoleArn	服務角色的 Amazon Resource Name (ARN) , 該角色會擴展對 AWS 資源的有限存取。 Type (類型) : string 必要 : 是
RoleSessionName	描述使用角色登入資料的工作階段名稱。 Type (類型) : string 必要 : 否

AttributeValue

在[Player](#)屬性鍵值對中使用這些值。此物件可讓您使用任何有效的資料類型來指定屬性值：字串、數字、字串陣列或資料映射。每個AttributeValue物件只能使用其中一個可用的屬性。

屬性	描述
attrType	指定屬性值的類型。 類型：AttrType 列舉 值。 必要：否
S	代表字串屬性值。 Type (類型)：string 必要：是
N	代表數值屬性值。 Type (類型)：double 必要：是
SL	代表字串屬性值的陣列。 Type (類型)：string[] 必要：是
SDM	代表字串索引鍵和雙值的字典。 Type (類型)：Dictionary<string, double> 必要：是

AwsStringOutcome

此資料類型來自 `動作`，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : string 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

GenericOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

DescribePlayerSessionsOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “DescribePlayerSessionsResult” 必要：否
成功	動作是否成功。 Type (類型) : bool 必要：是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要：否

DescribePlayerSessionsResult

屬性	描述
NextToken	表示結果下一頁開始的字符。若要指定結果集的開頭，請不要提供值。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : string 必要：是
PlayerSessions	包含符合請求之每個玩家工作階段屬性的物件集合。 Type (類型) : IList<the section called “PlayerSession”>

屬性	描述
	必要：
成功	動作是否成功。 Type (類型) : bool 必要：是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要：否

PlayerSession

屬性	描述
CreationTime	Type (類型) : long 必要：是
FleetId	Type (類型) : string 必要：是
GameSessionId	Type (類型) : string 必要：是
IpAddress	Type (類型) : string 必要：是
PlayerData	Type (類型) : string 必要：是
PlayerId	Type (類型) : string

屬性	描述
	必要：是
PlayerSessionId	Type (類型) : string 必要：是
連線埠	Type (類型) : int 必要：是
Status	類型：PlayerSessionStatus 列舉 。 必要：是
TerminationTime	Type (類型) : long 必要：是
DnsName	Type (類型) : string 必要：是

StartMatchBackfillOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “StartMatchBackfillResult” 必要：否
成功	動作是否成功。 Type (類型) : bool 必要：是

屬性	描述
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

StartMatchBackfillResult

屬性	描述
TicketId	Type (類型) : string 必要 : 是

GetComputeCertificateOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “GetComputeCertificateResult” 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

GetComputeCertificateResult

運算上 TLS 憑證的路徑和運算的主機名稱。

屬性	描述
CertificatePath	Type (類型) : string 必要 : 是
ComputeName	Type (類型) : string 必要 : 是

GetFleetRoleCredentialsOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “GetFleetRoleCredentialsResult” 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要 : 否

GetFleetRoleCredentialsResult

屬性	描述
AccessKeyId	要驗證並提供 AWS 資源存取權的存取金鑰 ID。 Type (類型) : string 必要 : 否
AssumedRoleId	服務角色所屬的使用者 ID。 Type (類型) : string 必要 : 否
AssumedRoleUserArn	服務角色所屬使用者的 Amazon Resource Name (ARN)。 Type (類型) : string 必要 : 否
過期	工作階段登入資料過期前的時間長度。 Type (類型) : DateTime 必要 : 否
SecretAccessKey	用於身分驗證的私密存取金鑰 ID。 Type (類型) : string 必要 : 否
SessionToken	識別目前作用中工作階段與 AWS 資源互動的字符。 Type (類型) : string 必要 : 否
成功	動作是否成功。 Type (類型) : bool

屬性	描述
	必要：是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要：否

AwsDateTimeOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : DateTime 必要：否
成功	動作是否成功。 Type (類型) : bool 必要：是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “GameLiftError” 必要：否

GameLiftError

屬性	描述
ErrorType	錯誤類型。

屬性	描述
	類型：GameLiftErrorType 列舉 。 必要：否
ErrorMessage	錯誤的名稱。 Type (類型)：string 必要：否
ErrorMessage	錯誤訊息。 Type (類型)：string 必要：否

列舉

為適用於 Amazon GameLift Servers(C#) 的伺服器 SDK 定義的列舉定義如下：

AttrType

- NONE
- STRING
- DOUBLE
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

指出錯誤類型的字串值。有效值包含：

- SERVICE_CALL_FAILED – 呼叫 AWS 服務失敗。
- LOCAL_CONNECTION_FAILED – 與的本機連線Amazon GameLift Servers失敗。
- NETWORK_NOT_INITIALIZED – 網路尚未初始化。
- GAMESESSION_ID_NOT_SET – 尚未設定遊戲工作階段 ID。
- BAD_REQUEST_EXCEPTION
- INTERNAL_SERVICE_EXCEPTION

- `ALREADY_INITIALIZED` – Amazon GameLift Servers 伺服器或用戶端已使用 `Initialize()` 初始化。
- `FLEET_MISMATCH` – 目標機群與 `gameSession` 或 `playerSession` 的機群不相符。
- `GAMELIFT_CLIENT_NOT_INITIALIZED` – Amazon GameLift Servers 用戶端尚未初始化。
- `GAMELIFT_SERVER_NOT_INITIALIZED` – Amazon GameLift Servers 伺服器尚未初始化。
- `GAME_SESSION_ENDED_FAILED` – 的 伺服器開發套件 Amazon GameLift Servers 無法聯絡 服務以報告遊戲工作階段已結束。
- `GAME_SESSION_NOT_READY` – Amazon GameLift Servers 伺服器遊戲工作階段未啟用。
- `GAME_SESSION_READY_FAILED` – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務來報告遊戲工作階段已就緒。
- `INITIALIZATION_MISMATCH` – 在 `Server : : Initialize()` 之後呼叫用戶端方法，反之亦然。
- `NOT_INITIALIZED` – Amazon GameLift Servers 伺服器或用戶端尚未使用 `Initialize()` 初始化。
- `NO_TARGET_ALIASID_SET` – 尚未設定目標 `aliasId`。
- `NO_TARGET_FLEET_SET` – 尚未設定目標機群。
- `PROCESS_ENDING_FAILED` – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務來報告程序即將結束。
- `PROCESS_NOT_ACTIVE` – 伺服器程序尚未啟用、未繫結至 `GameSession`，且無法接受或處理 `PlayerSessions`。
- `PROCESS_NOT_READY` – 伺服器程序尚未準備好啟用。
- `PROCESS_READY_FAILED` – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務以回報程序已就緒。
- `SDK_VERSION_DETECTION_FAILED` – SDK 版本偵測失敗。
- `STX_CALL_FAILED` – 呼叫 `XStx` 伺服器後端元件失敗。
- `STX_INITIALIZATION_FAILED` – `XStx` 伺服器後端元件無法初始化。
- `UNEXPECTED_PLAYER_SESSION` – 伺服器遇到未註冊的玩家工作階段。
- `WEBSOCKET_CONNECT_FAILURE`
- `WEBSOCKET_CONNECT_FAILURE_FORBIDDEN`
- `WEBSOCKET_CONNECT_FAILURE_INVALID_URL`
- `WEBSOCKET_CONNECT_FAILURE_TIMEOUT`
- `WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE` – 將訊息傳送至 GameLift Service WebSocket 的可重試失敗。

- WEBSOCKET_SEND_MESSAGE_FAILURE – 無法傳送訊息至 GameLift Service WebSocket。
- MATCH_BACKFILL_REQUEST_VALIDATION – 驗證請求失敗。
- PLAYER_SESSION_REQUEST_VALIDATION – 驗證請求失敗。

PlayerSessionCreationPolicy

字串值代表遊戲工作階段是否可接受新玩家。有效值包含：

- ACCEPT_ALL – 接受所有新玩家工作階段。
- DENY_ALL – 拒絕所有新玩家工作階段。
- NOT_SET – 遊戲工作階段未設定為接受或拒絕新的玩家工作階段。

PlayerSessionStatus

- ACTIVE
- COMPLETED (已完成)
- NOT_SET
- RESERVED
- 逾時

適用於 -- 資料類型的 C# 伺服器 SDK Amazon GameLift Servers 5.x

主題

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)

- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Destroy\(\)](#)

GetSdkVersion()

傳回內建至伺服器程序的目前開發套件版本編號。

語法

```
AwsStringOutcome GetSdkVersion();
```

傳回值

如果成功，將目前開發套件版本以 [the section called “AwsStringOutcome”](#) 物件傳回。傳回的字串包含版本號碼（範例 5.0.0）。如果不成功，則會傳回錯誤訊息。

範例

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

InitSDK()

初始化受管 EC2 機群的 Amazon GameLift Servers SDK。在啟動時，先呼叫此方法，再 Amazon GameLift Servers 進行任何其他與相關的初始化。此方法會從主機環境讀取伺服器參數，以設定伺服器與服務之間的通訊 Amazon GameLift Servers。它使用等冪符記，因此您可以在失敗時安全地重試此呼叫。

語法

```
GenericOutcome InitSDK();
```

傳回值

如果成功，會傳回 InitSdkOutcome 物件，表示伺服器程序已準備好呼叫 [ProcessReady\(\)](#)。

範例

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK();
```

InitSDK()

初始化 Anywhere 機群的 Amazon GameLift Servers SDK。在啟動時，先呼叫此方法，再Amazon GameLift Servers進行任何其他與 相關的初始化。此方法需要明確的伺服器參數，才能設定伺服器和服務之間的通訊Amazon GameLift Servers。它使用等冪符記，因此您可以在失敗時安全地重試此呼叫。

語法

```
GenericOutcome InitSDK(ServerParameters serverParameters);
```

參數

[ServerParameters](#)

若要在 Amazon GameLift ServersAnywhere 機群上初始化遊戲伺服器，請使用下列資訊建構ServerParameters物件：

- 用來連線至遊戲伺服器的 WebSocket URL。
- 用於託管遊戲伺服器的程序 ID。
- 託管遊戲伺服器程序的運算 ID。
- 包含 Amazon GameLift ServersAnywhere 運算的Amazon GameLift Servers機群 ID。
- Amazon GameLift Servers 操作產生的授權字符。

傳回值

如果成功，會傳回 InitSdkOutcome 物件，表示伺服器程序已準備好呼叫 [ProcessReady\(\)](#)。

Note

如果部署到 Anywhere 機群的遊戲組建呼叫InitSDK()失敗，請檢查建立組建資源時使用的ServerSdkVersion參數。您必須將此值明確設定為正在使用的伺服器 SDK 版本。此參數的預設值為 4.x，不相容。若要解決此問題，請建立新的組建，並將其部署到新的機群。

範例

```
//Define the server parameters
string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
string processId = "PID1234";
string fleetId = "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
string hostId = "HardwareAnywhere";
string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
ServerParameters serverParameters =
    new ServerParameters(webSocketUrl, processId, hostId, fleetId, authToken);

//Call InitSDK to establish a local connection with the GameLift agent to enable
    further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift Servers 通知伺服器程序已準備好託管遊戲工作階段。叫用 後呼叫此方法 [InitSDK\(\)](#)。每個程序只能呼叫此方法一次。

語法

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

參數

[ProcessParameters](#)

ProcessParameters 物件會保留伺服器程序的相關資訊。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例說明 方法和委派函數實作。

```
// Set parameters and call ProcessReady
```

```
ProcessParameters processParams = new ProcessParameters(  
    this.OnStartGameSession,  
    this.OnProcessTerminate,  
    this.OnHealthCheck,  
    this.OnUpdateGameSession,  
    port,  
    new LogParameters(new List<string>()  
        // Examples of log and error files written by the game server  
        {  
            "C:\\\\game\\\\logs",  
            "C:\\\\game\\\\error"  
        }  
    )  
);  
GenericOutcome processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

ProcessEnding()

Amazon GameLift Servers 通知伺服器程序正在終止。在所有其他清除任務（包括關閉作用中的遊戲工作階段）之後，以及在終止程序之前，呼叫此方法。根據的結果ProcessEnding()，程序會以成功 (0) 或錯誤 (-1) 結束並產生機群事件。如果程序因錯誤而終止，產生的機群事件為 SERVER_PROCESS_TERMINATED_UNHEALTHY。

語法

```
GenericOutcome ProcessEnding()
```

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會先呼叫 ProcessEnding() 和 Destroy() 再以成功或錯誤結束程式碼終止伺服器程序。

```
GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();  
GameLiftServerAPI.Destroy();  
  
if (processEndingOutcome.Success)  
{  
    Environment.Exit(0);  
}  
else
```

```
{  
    Console.WriteLine("ProcessEnding() failed. Error: " +  
        processEndingOutcome.Error.ToString());  
    Environment.Exit(-1);  
}
```

ActivateGameSession()

Amazon GameLift Servers 通知伺服器程序已啟用遊戲工作階段，現在已準備好接收玩家連線。在所有遊戲工作階段初始化之後，應該呼叫此動作做為回 `onStartGameSession()` 呼函數的一部分。

語法

```
GenericOutcome ActivateGameSession()
```

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例顯示的是做為 `onStartGameSession()` 委派函式一部分的 `ActivateGameSession()` 受到呼叫。

```
void OnStartGameSession(GameSession gameSession)  
{  
    // game-specific tasks when starting a new game session, such as loading map  
    // When ready to receive players  
    GenericOutcome activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();  
}
```

UpdatePlayerSessionCreationPolicy()

更新目前遊戲工作階段的能力，以接受新的玩家工作階段。遊戲工作階段可設定為接受或拒絕所有新的玩家工作階段。

語法

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy  
    playerSessionPolicy)
```

參數

playerSessionPolicy

指出遊戲工作階段是否接受新玩家的字串值。

有效值包含：

- ACCEPT_ALL – 接受所有新玩家工作階段。
- DENY_ALL – 拒絕所有新玩家工作階段。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例設定目前遊戲工作階段的加入政策為可接受所有玩家。

```
GenericOutcome updatePlayerSessionPolicyOutcome =  
    GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

GetGameSessionId()

擷取作用中伺服器程序託管的遊戲工作階段 ID。

對於未使用遊戲工作階段啟用的閒置程序，呼叫會傳回 [the section called “GameLiftError”](#)。

語法

```
AwsStringOutcome GetGameSessionId()
```

傳回值

如果成功，則會把遊戲工作階段 ID 當成 [the section called “AwsStringOutcome”](#) 物件傳回。如果不成功，會傳回錯誤訊息。」

範例

```
AwsStringOutcome getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetTerminationTime()

若設有終止時間，即傳回伺服器程序排定關閉的時間。伺服器程序在收到來自的 `onProcessTerminate()` 呼後採取此動作Amazon GameLift Servers。Amazon GameLift `ServersonProcessTerminate()` 會呼叫，原因如下：

- 當伺服器程序回報運作狀態不佳或未回應時Amazon GameLift Servers。
- 在縮減規模事件期間終止執行個體時。
- 當執行個體因 [spot-instance 中斷](#)而終止時。

語法

```
AwsDateTimeOutcome GetTerminationTime()
```

傳回值

如果成功，會將終止時間傳回為 [the section called “AwsDateTimeOutcome”](#) 物件。此值為終止時間，以自以來經過的刻度表示0001 00:00:00。例如，日期時間值2020-09-13 12:26:40 -000Z等於6373559680000000000刻度。如果沒有可用的終止時間，會傳回錯誤訊息。

範例

```
AwsDateTimeOutcome getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift Servers 通知具有指定玩家工作階段 ID 的玩家已連線至伺服器程序，且需要驗證。Amazon GameLift Servers 會驗證玩家工作階段 ID 是否有效。驗證玩家工作階段後，會將玩家位置 Amazon GameLift Servers的狀態從 RESERVED 變更為 ACTIVE。

語法

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

參數

playerSessionId

建立新的玩家工作階段時，GameLift 發行的唯一 ID。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例展示了用來處理連線請求的函數，其處理過程包括驗證和拒絕無效的玩家工作階段 ID。

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerSessionId)
{
    GenericOutcome acceptPlayerSessionOutcome =
    GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
    if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);
    }
}
```

RemovePlayerSession()

Amazon GameLift Servers 通知玩家已中斷與伺服器程序的連線。作為回應，Amazon GameLift Servers 會將玩家位置變更為可用。

語法

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

參數

playerSessionId

建立新玩家工作階段Amazon GameLift Servers時由 發出的唯一 ID。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
GenericOutcome removePlayerSessionOutcome =  
    GameLiftServerAPI.RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

擷取玩家工作階段資料，其中包括設定、工作階段中繼資料和玩家資料。使用此動作可取得單一玩家工作階段資訊、一個遊戲工作階段中所有玩家工作階段的資訊，或是與單一玩家 ID 關聯的所有玩家工作階段資訊。

語法

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest  
    describePlayerSessionsRequest)
```

參數

[DescribePlayerSessionsRequest](#)

描述要擷取哪些玩家工作階段的[the section called “DescribePlayerSessionsRequest”](#)物件。

傳回值

如果成功，會傳回[the section called “DescribePlayerSessionsOutcome”](#)物件，其中包含一組符合請求參數的玩家工作階段物件。

範例

此範例展示了讓所有玩家工作階段均主動連線至指定之遊戲工作階段的請求。透過省略 NextToken 並將限制值設定為 10，Amazon GameLift Servers 會傳回符合請求的前 10 個玩家工作階段記錄。

```
// Set request parameters  
DescribePlayerSessionsRequest describePlayerSessionsRequest = new  
    DescribePlayerSessionsRequest()  
{  
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    //gets the ID for the  
    current game session  
    Limit = 10,  
    PlayerSessionStatusFilter =  
        PlayerSessionStatusMapper.GetNameForPlayerSessionStatus(PlayerSessionStatus.ACTIVE)  
};
```

```
// Call DescribePlayerSessions
DescribePlayerSessionsOutcome describePlayerSessionsOutcome =
    GameLiftServerAPI.DescribePlayerSessions(describePlayerSessionsRequest);
```

StartMatchBackfill()

此動作會傳送請求，以便替 FlexMatch 所建立的遊戲工作階段開放空位找到新玩家。如需詳細資訊，請參閱[FlexMatch 回填功能](#)。

此為非同步動作。如果新玩家相符，會使用回呼函數 Amazon GameLift Servers 交付更新的配對建構器資料 `OnUpdateGameSession()`。

一個伺服器程序一次僅能有一個使用中的配對回填請求。若要發送新請求，請先呼叫 [StopMatchBackfill\(\)](#) 取消原始請求。

語法

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
    startBackfillRequest);
```

參數

[StartMatchBackfillRequest](#)

`StartMatchBackfillRequest` 物件會保留有關回填請求的資訊。

傳回值

傳回具有相符回填票證 ID 或失敗並顯示錯誤訊息的 [the section called "StartMatchBackfillOutcome"](#) 物件。

範例

```
// Build a backfill request
StartMatchBackfillRequest startBackfillRequest = new StartMatchBackfillRequest()
{
    TicketId = "1111aaaa-22bb-33cc-44dd-5555eeee66ff", //optional
    MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result, // gets ID for
    current game session
    MatchmakerData matchmakerData =
```

```
MatchmakerData.FromJson(gameSession.MatchmakerData), // gets matchmaker data for
current players
// get matchmakerData.Players
// remove data for players who are no longer connected
Players = ListOfPlayersRemainingInTheGame
};

// Send backfill request
StartMatchBackfillOutcome startBackfillOutcome =
    GameLiftServerAPI.StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void OnUpdateGameSession(GameSession myGameSession)
{
    // game-specific tasks to prepare for the newly matched players and update matchmaker
    data as needed
}
```

StopMatchBackfill()

取消作用中的配對回填請求。如需詳細資訊，請參閱[FlexMatch回填功能](#)。

語法

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

參數

[StopMatchBackfillRequest](#)

提供您正在停止之配對票證詳細資訊的StopMatchBackfillRequest物件。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
// Set backfill stop request parameters
StopMatchBackfillRequest stopBackfillRequest = new StopMatchBackfillRequest(){
    TicketId = "1111aaaa-22bb-33cc-44dd-5555eeee66ff", //optional, if not provided one is
    autogenerated
}
```

```
MatchmakingConfigurationArn = "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
GameSessionId = GameLiftServerAPI.GetGameSessionId().Result //gets the ID for the
current game session
};
GenericOutcome stopBackfillOutcome =
GameLiftServerAPI.StopMatchBackfillRequest(stopBackfillRequest);
```

GetComputeCertificate()

擷取用於加密遊戲伺服器與遊戲用戶端之間網路連線的 TLS 憑證路徑。您可以在向 Amazon GameLift Servers Anywhere 機群註冊運算裝置時使用憑證路徑。如需詳細資訊，請參閱 [RegisterCompute](#)。

語法

```
GetComputeCertificateOutcome GetComputeCertificate();
```

傳回值

傳回包含下列項目的 `GetComputeCertificateResponse` 物件：

- `CertificatePath`：運算資源上 TLS 憑證的路徑。使用 Amazon GameLift Servers 受管機群時，此路徑包含：
 - `certificate.pem`：最終使用者憑證。完整的憑證鏈是 `certificateChain.pem` 附加到此憑證的組合。
 - `certificateChain.pem`：包含根憑證和中繼憑證的憑證鏈。
 - `rootCertificate.pem`：根憑證。
 - `privateKey.pem`：最終使用者憑證的私有金鑰。
- `ComputeName`：運算資源的名稱。

範例

```
GetComputeCertificateOutcome getComputeCertificateOutcome =
GameLiftServerAPI.GetComputeCertificate();
```

GetFleetRoleCredentials()

擷取授權與其他互動的 IAM 角色登入 Amazon GameLift Servers 資料 AWS 服務。如需詳細資訊，請參閱 [與機群中的其他 AWS 資源通訊](#)。

語法

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest request);
```

參數

[GetFleetRoleCredentialsRequest](#)

角色登入資料，可將 AWS 資源的有限存取權延伸到遊戲伺服器。

傳回值

其會傳回 [the section called “GetFleetRoleCredentialsOutcome”](#) 物件。

範例

```
// form the fleet credentials request
GetFleetRoleCredentialsRequest getFleetRoleCredentialsRequest = new
    GetFleetRoleCredentialsRequest(){
        RoleArn = "arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction"
    };
GetFleetRoleCredentialsOutcome GetFleetRoleCredentialsOutcome credentials =
    GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Destroy()

從記憶體釋放Amazon GameLift Servers遊戲伺服器 SDK。最佳實務是在終止程序之後ProcessEnding()和之前呼叫此方法。如果您使用的是 Anywhere 機群，而且不是在每次遊戲工作階段後終止伺服器程序，請呼叫 Destroy()，然後在通知 Amazon GameLift Servers 程序已準備好使用託管遊戲工作階段之前InitSDK()重新初始化 ProcessReady()。

語法

```
GenericOutcome Destroy()
```

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
// Operations to end game sessions and the server process
GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();

// Shut down and destroy the instance of the GameLift Game Server SDK
GenericOutcome destroyOutcome = GameLiftServerAPI.Destroy();

// Exit the process with success or failure
if (processEndingOutcome.Success)
{
    Environment.Exit(0);
}
else
{
    Console.WriteLine("ProcessEnding() failed. Error: " +
        processEndingOutcome.Error.ToString());
    Environment.Exit(-1);
}
```

適用於 Amazon GameLift Servers -- 動作的 Go 伺服器 SDK

使用伺服器 SDK 5.x 參考整合多玩家遊戲，以便與 進行託管。Amazon GameLift Servers 如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

GameLiftServerAPI.go 定義 Go 伺服器 SDK 動作。

適用於 Amazon GameLift Servers -- 資料類型的 Go 伺服器 SDK

使用伺服器 SDK 參考來整合多玩家遊戲，以便與 進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

GameLiftServerAPI.go 定義 Go 伺服器 SDK 動作。

[適用於 Amazon GameLift Servers -- 動作的 Go 伺服器 SDK](#)

資料類型

- [LogParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)

- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)

LogParameters

識別在遊戲工作階段期間產生的檔案的物件，Amazon GameLift Servers 您想要在遊戲工作階段結束後上傳和存放。遊戲伺服器會將 Amazon GameLift Servers 作為 [ProcessReady\(\)](#) 呼叫中 `ProcessParameters` 物件的一部分 `LogParameters` 提供給。

屬性	Description
LogPaths	Amazon GameLift Servers 您要存放以供日後存取之遊戲伺服器日誌檔案的目錄路徑清單。伺服器程序會在每個遊戲工作階段期間產生這些檔案。您可以在遊戲伺服器中定義檔案路徑和名稱，並將其存放在根遊戲建置目錄中。 日誌路徑必須是絕對的。例如，如果您的遊戲組建將遊戲工作階段日誌儲存在如的路徑中 <code>MyGame\sessionLogs\</code>，則路徑會位於 Windows 執行個體 <code>c:\game\MyGame\sessionLogs</code> 上。 Type (類型) : []string 必要：否

ProcessParameters

描述伺服器程序與之間通訊的物件 Amazon GameLift Servers。伺服器程序 Amazon GameLift Servers 透過呼叫 將此資訊提供給 [ProcessReady\(\)](#)。

屬性	Description
LogParameters	具有在遊戲工作階段期間產生之檔案目錄路徑的物件。會 Amazon GameLift Servers 複製和存放檔案以供日後存取。

	Type (類型) : LogParameters 必要 : 否
OnHealthCheck	Amazon GameLift Servers 呼叫 從伺服器程序請求運作狀態報告的回呼函數。每 60 秒會Amazon GameLift Servers呼叫此函數，並等待 60 秒進行回應。FALSE 如果運作狀態TRUE不良，伺服器程序會傳回。如果未傳回任何回應，會將伺服器程序Amazon GameLift Servers記錄為運作狀態不佳。 Type (類型) : OnHealthCheck func() bool 必要 : 否
OnProcessTerminate	Amazon GameLift Servers 呼叫 以強制伺服器程序關閉的回呼函數。呼叫此函數後，會Amazon GameLift Servers等待 5 分鐘讓伺服器程序關閉，並在關閉伺服器程序之前回應ProcessEnding()呼叫。 Type (類型) : OnProcessTerminate func() 必要 : 是
OnStartGameSession	Amazon GameLift Servers 呼叫 以將更新後的遊戲工作階段物件傳遞至伺服器程序的回呼函數。會在處理配對回填請求時Amazon GameLift Servers呼叫此函數，以提供更新後的配對建構器資料。它會傳遞GameSession物件、狀態更新 (updateReason) 和相符回填票證 ID。 Type (類型) : OnStartGameSession func (model.GameSession) 必要 : 是
OnUpdateGameSession	Amazon GameLift Servers 叫用以將更新後的遊戲工作階段資訊傳遞至伺服器程序的回呼函數。會在處理配對回填請求後Amazon GameLift Servers呼叫此函數，以提供更新的配對建構器資料。 Type (類型) : OnUpdateGameSession func (model.UpdateGameSession) 必要 : 否

Port 伺服器程序接聽新玩家連線的連接埠號碼。值必須屬於為部署此遊戲伺服器組建之機群所設定的連接埠範圍。此連接埠號碼包含在遊戲工作階段和遊戲工作階段物件中，遊戲工作階段會使用該物件來連接到伺服器程序。

Type (類型) : int

必要：是

UpdateGameSession

遊戲工作階段物件的更新，其中包括遊戲工作階段更新的原因，以及如果使用回填來填充遊戲工作階段中的玩家工作階段，則相關的回填票證 ID。

屬性	Description
GameSession	GameSession 物件。GameSession 物件包含描述遊戲工作階段的屬性。 Type (類型) : GameSession GameSession() 必要：是
UpdateReason	正在更新遊戲工作階段的原因。 Type (類型) : UpdateReason UpdateReason() 必要：是
BackfillTicketId	嘗試更新遊戲工作階段的重填票證 ID。 Type (類型) : String 必要：否

GameSession

遊戲工作階段的詳細資訊。

屬性	Description
GameSessionId	遊戲工作階段的唯一識別符。遊戲工作階段 Amazon Resource Name (ARN) 的格式如下：<code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code>。 Type (類型) : String 必要：否
名稱	遊戲工作階段的描述性標籤。 Type (類型) : String 必要：否
FleetId	遊戲工作階段執行所在的機群的唯一識別符。 Type (類型) : String 必要：否
MaximumPlayerSessionCount	遊戲工作階段的玩家連線數目上限。 Type (類型) : Integer 必要：否
連線埠	遊戲工作階段的連接埠號碼。若要連線到Amazon GameLift Servers遊戲伺服器，應用程式需要 IP 地址和連接埠號碼。 Type (類型) : Integer 必要：否
IpAddress	遊戲工作階段的 IP 地址。若要連線到Amazon GameLift Servers遊戲伺服器，應用程式需要 IP 地址和連接埠號碼。 Type (類型) : String 必要：否

屬性	Description
GameSessionData	一組自訂遊戲工作階段屬性，格式為單一字串值。 Type (類型) : String 必要 : 否
MatchmakerData	用於建立遊戲工作階段的配對程序資訊，以 JSON 語法格式化為字串。除了使用的配對組態之外，還包含指派給配對的所有玩家的資料，包括玩家屬性和團隊指派。 Type (類型) : String 必要 : 否
GameProperties	一組遊戲工作階段的自訂屬性，格式為 key : value 對。這些屬性會與啟動新遊戲工作階段的請求一起傳遞。 Type (類型) : map[string] string 必要 : 否
DnsName	指派給執行遊戲工作階段之執行個體的 DNS 識別符。值的格式如下： • 啟用 TLS 的機群 : <unique identifier>.<region identifier>.amazongamelift.com 。 • Non-TLS-enabled 的機群 : ec2-<unique identifier>.compute.amazonaws.com 。 連線至在啟用 TLS 的機群上執行的遊戲工作階段時，您必須使用 DNS 名稱，而非 IP 地址。 Type (類型) : String 必要 : 否

ServerParameters

用來維護 Amazon GameLift ServersAnywhere 伺服器與 Amazon GameLift Servers服務之間連線的資訊。使用 啟動新的伺服器程序時，會使用此資訊[InitSDK\(\)](#)。對於託管在Amazon GameLift Servers受管 EC2 執行個體上的伺服器，請使用空物件。

屬性	Description
WebSocket URL	當您RegisterCompute 使用 Amazon GameLift ServersAnywhere 運算資源時，會GameLiftServerSdkEndpoint Amazon GameLift Servers傳回。 Type (類型) : string 必要：是
ProcessID	向託管遊戲的伺服器程序註冊的唯一識別符。 Type (類型) : string 必要：是
HostID	託管新伺服器程序之運算資源的唯一識別符。 HostID 是當您註冊運算ComputeName 時使用的。如需詳細資訊，請參閱RegisterCompute。 Type (類型) : string 必要：是
FleetID	運算註冊目的地機群的唯一識別符。如需詳細資訊，請參閱 RegisterCompute。 Type (類型) : string 必要：是
AuthToken	產生的身分驗證字符Amazon GameLift Servers，用於向 驗證您的伺服器Amazon GameLift Servers。如需詳細資訊，請參閱 GetComputeAuthToken。 Type (類型) : string

屬性	Description
	必要：是

StartMatchBackfillRequest

用來建立配對回填請求的資訊。遊戲伺服器會在[StartMatchBackfill\(\)](#)呼叫Amazon GameLift Servers中將此資訊傳達給。

屬性	Description
GameSessionArn	唯一的遊戲工作階段識別符。API 操作會以 ARN 格式GetGameSessionId 傳回識別符。 Type (類型) : String 必要：是
MatchmakingConfigurationArn	配對建構器用於此請求的唯一識別符（以 ARN 的形式）。原始遊戲工作階段的配對建構器 ARN 位於配對建構器資料屬性中的遊戲工作階段物件中。如需配對建構器資料的詳細資訊，請參閱使用配對建構器資料。 Type (類型) : String 必要：是
Players	一組資料，代表目前在遊戲工作階段中的所有玩家。配對建構器使用此項資訊搜尋適合配對現有玩家的新玩家。 Type (類型) : []model.Player 必要：是
TicketId	配對或配對回填請求票證的唯一識別符。如果您不提供值，Amazon GameLift Servers會產生值。您可使用此識別項依據需求追蹤配對回填票證狀態或取消要求。 Type (類型) : String 必要：否

Player

代表配對中玩家的物件。當配對請求開始時，玩家會有玩家 ID、屬性和可能的延遲資料。會在配對完成後Amazon GameLift Servers新增團隊資訊。

屬性	Description
LatencyInMS	一組以毫秒為單位的值，指出玩家連線到位置時所經歷的延遲量。 如果使用此屬性，則播放器只會符合列出的位置。若配對構建器有評估玩家延遲的規則，玩家則必須回報延遲度，方可配對。 Type (類型) : map[string] int 必要：否
PlayerAttributes	包含玩家資訊的金鑰：值對集合，可用於配對。玩家屬性金鑰必須符合配對規則集中使用的 PlayerAttributes。 如需玩家屬性的詳細資訊，請參閱 AttributeValue。 Type (類型) : map[string] AttributeValue 必要：否
PlayerId	玩家的唯一識別符。 Type (類型) : String 必要：否
團隊	玩家在配對中獲指派的隊伍名稱。您可以在配對規則集中定義團隊名稱。 Type (類型) : String 必要：否

DescribePlayerSessionsRequest

指定要擷取哪些玩家工作階段的物件。伺服器程序會在[DescribePlayerSessions\(\)](#)呼叫時提供此資訊Amazon GameLift Servers。

屬性	Description
GameSessionID	唯一的遊戲工作階段識別符。請使用此參數要求特定遊戲工作階段的所有玩家工作階段。 遊戲工作階段 ID 格式為 <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code> 。GameSessionID 是自訂 ID 字串或產生的字串。 Type (類型) : String 必要 : 否
PlayerSessionID	玩家工作階段的唯一識別符。使用此參數請求單一特定玩家工作階段。 Type (類型) : String 必要 : 否
PlayerID	玩家的唯一識別符。使用此參數來請求特定玩家的所有玩家工作階段。請參閱 產生玩家 IDs。 Type (類型) : String 必要 : 否
PlayerSessionStatusFilter	要篩選結果的玩家工作階段狀態。可能的玩家工作階段狀態包括： • RESERVED – 已收到玩家工作階段請求，但玩家尚未連線到伺服器程序或已經過驗證。 • ACTIVE – 播放器已由伺服器程序驗證並連線。 • 已完成 – 玩家連線中斷。 • TIMEDOUT – 已收到玩家工作階段請求，但玩家未連線或未在逾時限制 (60 秒) 內驗證。 Type (類型) : String 必要 : 否

屬性	Description
NextToken	表示結果下一頁開始的字符。若要指定結果集的開頭，請不要提供值。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : String 必要 : 否
Limit	回傳結果的數量上限。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : int 必要 : 否

StopMatchBackfillRequest

用來取消配對回填請求的資訊。遊戲伺服器會在[StopMatchBackfill\(\)](#) 通話中將此資訊傳達給 Amazon GameLift Servers 服務。

屬性	Description
GameSessionArn	取消請求的唯一遊戲工作階段識別符。 Type (類型) : string 必要 : 否
MatchmakingConfigurationArn	此請求傳送至的配對建構器的唯一識別符。 Type (類型) : string 必要 : 否
TicketId	要取消之回填請求票證的唯一識別符。 Type (類型) : string 必要 : 否

GetFleetRoleCredentialsRequest

角色登入資料，可將 AWS 資源的有限存取權延伸到遊戲伺服器。如需詳細資訊，請參閱 [設定的 IAM 服務角色 Amazon GameLift Servers](#)。

屬性	Description
RoleArn	服務角色的 ARN，可延伸對 AWS 資源的有限存取。 Type (類型) : string 必要：是
RoleSessionName	描述使用角色登入資料的工作階段名稱。 Type (類型) : string 必要：是

[適用於 Amazon GameLift Servers -- 資料類型的 Go 伺服器 SDK](#)

動作

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)

- [GetFleetRoleCredentials\(\)](#)
- [Destroy\(\)](#)

GetSdkVersion()

傳回內建至伺服器程序的目前開發套件版本編號。

語法

```
func GetSdkVersion() (string, error)
```

傳回值

如果成功，會以字串的形式傳回目前的 SDK 版本。傳回的字串包含版本編號（範例 5.0.0）。如果不成功，會傳回錯誤訊息，例如 `common.SdkVersionDetectionFailed`。

範例

```
version, err := server.GetSdkVersion()
```

InitSDK()

初始化 Amazon GameLift Servers 開發套件。在與相關的任何其他初始化 Amazon GameLift Servers 發生之前，請在啟動時呼叫此方法。此方法會設定伺服器和服務之間的通訊 Amazon GameLift Servers。

語法

```
func InitSDK(params ServerParameters) error
```

參數

[ServerParameters](#)

若要在 Amazon GameLift Servers Anywhere 機群上初始化遊戲伺服器，請使用下列資訊建構 `ServerParameters` 物件：

- 用來連線至遊戲伺服器的 WebSocket URL。
- 用於託管遊戲伺服器的程序 ID。
- 託管遊戲伺服器程序的運算 ID。

- 包含 Amazon GameLift ServersAnywhere 運算的Amazon GameLift Servers機群 ID。
- Amazon GameLift Servers 操作產生的授權字符。

若要在Amazon GameLift Servers受管 EC2 機群上初始化遊戲伺服器，請建構沒有參數的ServerParameters物件。透過此呼叫，Amazon GameLift Servers代理程式會設定運算環境，並自動為您連線至 Amazon GameLift Servers服務。

傳回值

如果成功，會傳回nil錯誤，指出伺服器程序已準備好呼叫 [ProcessReady\(\)](#)。

Note

如果部署到 Anywhere 機群的遊戲組建呼叫InitSDK()失敗，請檢查建立組建資源時使用的ServerSdkVersion參數。您必須將此值明確設定為正在使用的伺服器 SDK 版本。此參數的預設值為 4.x，不相容。若要解決此問題，請建立新的組建，並將其部署到新的機群。

範例

Amazon GameLift ServersAnywhere 範例

```
//Define the server parameters
serverParameters := ServerParameters {
    WebSocketURL: "wss://us-west-1.api.amazongamelift.com",
    ProcessID: "PID1234",
    HostID: "HardwareAnywhere",
    FleetID: "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa",
    AuthToken: "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
}

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
err := server.InitSDK(serverParameters)
```

Amazon GameLift Servers 受管 EC2 範例

```
//Define the server parameters
serverParameters := ServerParameters {}
```

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
err := server.InitSDK(serverParameters)
```

ProcessReady()

Amazon GameLift Servers 通知伺服器程序已準備好託管遊戲工作階段。叫用 後呼叫此方法[InitSDK\(\)](#)。每個程序只能呼叫此方法一次。

語法

```
func ProcessReady(param ProcessParameters) error
```

參數

ProcessParameters

[ProcessParameters](#) 物件會傳達下列伺服器程序的相關資訊：

- 在遊戲伺服器程式碼中實作的回呼方法名稱，Amazon GameLift Servers服務會叫用這些方法來與伺服器程序通訊。
- 伺服器程序正在接聽的連接埠號碼。
- 包含Amazon GameLift Servers您要擷取和儲存之任何遊戲工作階段特定檔案路徑的[LogParameters](#)資料類型。

傳回值

如果方法失敗，則傳回包含錯誤訊息的錯誤。nil 如果方法成功，則傳回。

範例

此範例會說明 [ProcessReady\(\)](#) 呼叫和委派函數的實作。

```
// Define the process parameters
processParams := ProcessParameters {
    OnStartGameSession: gameProcess.OnStartGameSession,
    OnUpdateGameSession: gameProcess.OnGameSessionUpdate,
    OnProcessTerminate: gameProcess.OnProcessTerminate,
    OnHealthCheck: gameProcess.OnHealthCheck,
    Port: port,
    LogParameters: LogParameters {    // logging and error example
        []string {"C:\\game\\logs", "C:\\game\\error"}
```

```
}  
}  
  
err := server.ProcessReady(processParams)
```

ProcessEnding()

Amazon GameLift Servers 通知伺服器程序正在終止。在所有其他清除任務（包括關閉作用中的遊戲工作階段）之後，以及在終止程序之前，呼叫此方法。根據的結果 `ProcessEnding()`，程序會以成功 (0) 或錯誤 (-1) 結束並產生機群事件。如果程序因錯誤而終止，產生的機群事件為 `SERVER_PROCESS_TERMINATED_UNHEALTHY`。

語法

```
func ProcessEnding() error
```

傳回值

傳回 0 錯誤代碼或定義的錯誤代碼。

範例

```
// operations to end game sessions and the server process  
defer func() {  
    err := server.ProcessEnding()  
    server.Destroy()  
    if err != nil {  
        fmt.Println("ProcessEnding() failed. Error: ", err)  
        os.Exit(-1)  
    } else {  
        os.Exit(0)  
    }  
}
```

ActivateGameSession()

Amazon GameLift Servers 通知伺服器程序已啟用遊戲工作階段，現在已準備好接收玩家連線。在所有遊戲工作階段初始化後，此動作會呼叫為 `onStartGameSession()` 函數的一部分。

語法

```
func ActivateGameSession() error
```

傳回值

如果方法失敗，則傳回包含錯誤訊息的錯誤。

範例

此範例顯示`ActivateGameSession()`稱為`onStartGameSession()`委派函數的一部分。

```
func OnStartGameSession(GameSession gameSession) {  
    // game-specific tasks when starting a new game session, such as loading map  
    // Activate when ready to receive players  
    err := server.ActivateGameSession();  
}
```

UpdatePlayerSessionCreationPolicy()

更新目前遊戲工作階段的能力，以接受新的玩家工作階段。遊戲工作階段可設定為接受或拒絕所有新的玩家工作階段。

語法

```
func UpdatePlayerSessionCreationPolicy(policy model.PlayerSessionCreationPolicy) error
```

參數

playerSessionCreationPolicy

指出遊戲工作階段是否接受新玩家的字串值。

有效值包含：

- **model.AcceptAll** – 接受所有新的玩家工作階段。
- **model.DenyAll** – 拒絕所有新的玩家工作階段。

傳回值

如果發生失敗，則傳回包含錯誤訊息的錯誤。

範例

此範例設定目前遊戲工作階段的加入政策為可接受所有玩家。

```
err := server.UpdatePlayerSessionCreationPolicy(model.AcceptAll)
```

GetGameSessionId()

擷取作用中伺服器程序託管的遊戲工作階段 ID。

語法

```
func GetGameSessionID() (string, error)
```

參數

此動作沒有參數。

傳回值

如果成功，會傳回遊戲工作階段 ID 和 nil 錯誤。對於尚未透過遊戲工作階段啟用的閒置程序，呼叫會傳回空字串和 nil 錯誤。

範例

```
gameSessionID, err := server.GetGameSessionID()
```

GetTerminationTime()

傳回伺服器程序排程在可用終止時間時關閉的時間。伺服器程序在收到來自的 `onProcessTerminate()` 呼後會採取此動作 Amazon GameLift Servers。Amazon GameLift `ServersonProcessTerminate()` 會呼叫，原因如下：

- 當伺服器程序回報運作狀態不佳或尚未回應時 Amazon GameLift Servers。
- 在縮減規模事件期間終止執行個體時。
- 當執行個體因 [spot-instance 中斷](#) 而終止時。

語法

```
func GetTerminationTime() (int64, error)
```

傳回值

如果成功，會以 epoch 秒傳回排定關閉伺服器程序的時間戳記，並終止 nil 錯誤。值是終止時間，以經過的刻度表示 0001 00:00:00。例如，日期時間值 2020-09-13 12:26:40 -0000Z 等於 6373559680000000000 刻度。如果沒有可用的終止時間，會傳回錯誤訊息。

範例

```
terminationTime, err := server.GetTerminationTime()
```

AcceptPlayerSession()

Amazon GameLift Servers 通知具有指定玩家工作階段 ID 的玩家已連線至伺服器程序，且需要驗證。Amazon GameLift Servers 會驗證玩家工作階段 ID 是否有效。驗證玩家工作階段後，會將玩家位置 Amazon GameLift Servers 的狀態從 RESERVED 變更為 ACTIVE。

語法

```
func AcceptPlayerSession(playerSessionID string) error
```

參數

playerSessionId

建立新玩家工作階段 Amazon GameLift Servers 時由發出的唯一 ID。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會處理連線請求，其中包含驗證和拒絕非有效的玩家工作階段 IDs。

```
func ReceiveConnectingPlayerSessionID(conn Connection, playerSessionID string) {  
    err := server.AcceptPlayerSession(playerSessionID)  
    if err != nil {  
        connection.Accept()  
    } else {  
        connection.Reject(err.Error())  
    }  
}
```

RemovePlayerSession()

Amazon GameLift Servers 通知玩家已中斷與伺服器程序的連線。作為回應，Amazon GameLift Servers 會將玩家位置變更為可用。

語法

```
func RemovePlayerSession(playerSessionID string) error
```

參數

playerSessionId

建立新玩家工作階段Amazon GameLift Servers時由 發出的唯一 ID。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
err := server.RemovePlayerSession(playerSessionID)
```

DescribePlayerSessions()

擷取玩家工作階段資料，其中包括設定、工作階段中繼資料和玩家資料。使用此方法取得下列相關資訊：

- 單一玩家工作階段
- 遊戲工作階段中的所有玩家工作階段
- 與單一玩家 ID 相關聯的所有玩家工作階段

語法

```
func DescribePlayerSessions(req request.DescribePlayerSessionsRequest)
(result.DescribePlayerSessionsResult, error) {
    return srv.describePlayerSessions(&req)
}
```

參數

[DescribePlayerSessionsRequest](#)

DescribePlayerSessionsRequest 物件說明要擷取的玩家工作階段。

傳回值

如果成功，會傳回DescribePlayerSessionsResult物件，其中包含一組符合請求參數的玩家工作階段物件。

範例

此範例會請求主動連線到指定遊戲工作階段的所有玩家工作階段。透過省略 NextToken 並將限制值設定為 10，會Amazon GameLift Servers傳回符合請求的前 10 個玩家工作階段記錄。

```
// create request
describePlayerSessionsRequest := request.NewDescribePlayerSessions()
describePlayerSessionsRequest.GameSessionID, _ = server.GetGameSessionID() // get ID
for the current game session
describePlayerSessionsRequest.Limit = 10 // return the
first 10 player sessions
describePlayerSessionsRequest.PlayerSessionStatusFilter = "ACTIVE" // Get all
player sessions actively connected to the game session

describePlayerSessionsResult, err :=
server.DescribePlayerSessions(describePlayerSessionsRequest)
```

StartMatchBackfill()

此動作會傳送請求，以便替 FlexMatch 所建立的遊戲工作階段開放空位找到新玩家。如需詳細資訊，請參閱[FlexMatch回填功能](#)。

此為非同步動作。如果新玩家相符，會使用回呼函數 Amazon GameLift Servers交付更新的配對建構器資料OnUpdateGameSession()。

一個伺服器程序一次僅能有一個使用中的配對回填請求。若要發送新請求，請先呼叫[StopMatchBackfill\(\)](#) 取消原始請求。

語法

```
func StartMatchBackfill(req request.StartMatchBackfillRequest)
(result.StartMatchBackfillResult, error)
```

參數

[StartMatchBackfillRequest](#)

StartMatchBackfillRequest 物件會傳達下列資訊：

- 指派給回填請求的票證 ID。此資訊是選用的；如果未提供 ID，則 Amazon GameLift Servers 會產生一個 ID。
- 傳送請求對象的配對建構器。必須填入完整的組態 ARN。此值位於遊戲工作階段的配對建構器資料中。
- 要回填的遊戲工作階段 ID。
- 遊戲工作階段目前玩家的可用配對資料。

傳回值

傳回具有相符回填票證 ID 或失敗並顯示錯誤訊息的 `StartMatchBackfillResult` 物件。

範例

```
// form the request
startBackfillRequest := request.NewStartMatchBackfill()
startBackfillRequest.RequestID = "1111aaaa-22bb-33cc-44dd-5555eeee66ff" // optional
startBackfillRequest.MatchmakingConfigurationArn = "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"
var matchMaker model.MatchmakerData
if err := matchMaker.UnmarshalJSON([]byte(gameSession.MatchmakerData)); err != nil {

    return
}
startBackfillRequest.Players = matchMaker.Players
res, err := server.StartMatchBackfill(startBackfillRequest)

// Implement callback function for backfill
func OnUpdateGameSession(myGameSession model.GameSession) {
    // game-specific tasks to prepare for the newly matched players and update
    matchmaker data as needed
}
```

StopMatchBackfill()

取消作用中的配對回填請求。如需詳細資訊，請參閱[FlexMatch 回填功能](#)。

語法

```
func StopMatchBackfill(req request.StopMatchBackfillRequest) error
```

參數

StopMatchBackfillRequest

StopMatchBackfillRequest 物件，可識別要取消的配對票證：

- 指派給回填請求的票證 ID。
- 回填請求傳送到的配對建構器。
- 與回填請求相關聯的遊戲工作階段。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
stopBackfillRequest := request.NewStopMatchBackfill() // Use this function to create request
stopBackfillRequest.TicketID = "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
stopBackfillRequest.MatchmakingConfigurationArn = "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"

//error
err := server.StopMatchBackfill(stopBackfillRequest)
```

GetComputeCertificate()

擷取用於加密遊戲伺服器與遊戲用戶端之間網路連線的 TLS 憑證路徑。您可以在向 Amazon GameLift ServersAnywhere 機群註冊運算裝置時使用憑證路徑。如需詳細資訊，請參閱 [RegisterCompute](#)。

語法

```
func GetComputeCertificate() (result.GetComputeCertificateResult, error)
```

傳回值

傳回包含下列項目的GetComputeCertificateResult物件：

- CertificatePath：運算資源上 TLS 憑證的路徑。使用 Amazon GameLift Servers受管機群時，此路徑包含：

- `certificate.pem`：最終使用者憑證。完整的憑證鏈是`certificateChain.pem`附加到此憑證的組合。
- `certificateChain.pem`：包含根憑證和中繼憑證的憑證鏈。
- `rootCertificate.pem`：根憑證。
- `privateKey.pem`：最終使用者憑證的私有金鑰。
- `ComputeName`：運算資源的名稱。

範例

```
tlsCertificate, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

GetFleetRoleCredentials()

擷取您建立的服務角色登入資料，將許可擴展至其他 AWS 服務 Amazon GameLift Servers。這些登入資料可讓您的遊戲伺服器使用您的 AWS 資源。如需詳細資訊，請參閱[設定的 IAM 服務角色 Amazon GameLift Servers](#)。

語法

```
func GetFleetRoleCredentials(  
    req request.GetFleetRoleCredentialsRequest,  
) (result.GetFleetRoleCredentialsResult, error) {  
    return srv.getFleetRoleCredentials(&req)  
}
```

參數

[GetFleetRoleCredentialsRequest](#)

將 AWS 資源的有限存取權延伸到遊戲伺服器的角色登入資料。

傳回值

傳回包含下列項目的`GetFleetRoleCredentialsResult`物件：

- `AssumedRoleUserArn` - 服務角色所屬使用者的 Amazon Resource Name (ARN)。
- `AssumedRoleId` - 服務角色所屬的使用者 ID。

- `AccessKeyId` - 用於驗證和提供 AWS 資源存取權的存取金鑰 ID。
- `SecretAccessKey` - 用於身分驗證的私密存取金鑰 ID。
- `SessionToken` - 識別目前作用中工作階段與 AWS 資源互動的權杖。
- 過期 - 工作階段登入資料過期前的時間長度。

範例

```
// form the customer credentials request
getFleetRoleCredentialsRequest := request.NewGetFleetRoleCredentials()
getFleetRoleCredentialsRequest.RoleArn = "arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction"

credentials, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

Destroy()

從記憶體釋放 Amazon GameLift Servers 遊戲伺服器 SDK。最佳實務是在終止程序之後 `ProcessEnding()` 和之前呼叫此方法。如果您使用的是 Anywhere 機群，而且沒有在每次遊戲工作階段後終止伺服器程序，請呼叫 `Destroy()`，然後在通知 Amazon GameLift Servers 程序已準備好使用託管遊戲工作階段之前 `InitSDK()` 重新初始化 `ProcessReady()`。

語法

```
func Destroy() error {
    return srv.destroy()
}
```

傳回值

如果方法失敗，則傳回包含錯誤訊息的錯誤。

範例

```
// operations to end game sessions and the server process
defer func() {
    err := server.ProcessEnding()
    server.Destroy()
    if err != nil {
        fmt.Println("ProcessEnding() failed. Error: ", err)
        os.Exit(-1)
    }
}
```

```
} else {  
    os.Exit(0)  
}  
}
```

適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x

使用 Amazon GameLift Servers Unreal 伺服器 SDK 5.x 參考，協助您準備多玩家遊戲以搭配使用。Amazon GameLift Servers 如需整合程序的詳細資訊，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。如果您使用 Unreal 的 Amazon GameLift Servers 外掛程式，另請參閱 [Amazon GameLift Servers Unreal Engine 的外掛程式](#)。

Note

本主題說明您可以在為 Unreal Engine 建置時使用的 Amazon GameLift Servers C++ API。具體而言，本文件適用於您使用 `-DBUILD_FOR_UNREAL=1` 選項編譯的程式碼。

適用於 -- 資料類型的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x

使用 Amazon GameLift Servers Unreal 伺服器 SDK 5.x 參考，協助您準備多玩家遊戲以搭配使用。Amazon GameLift Servers 如需整合程序的詳細資訊，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。如果您使用 Unreal 的 Amazon GameLift Servers 外掛程式，另請參閱 [Amazon GameLift Servers Unreal Engine 的外掛程式](#)。

Note

本主題說明您可以在為 Unreal Engine 建置時使用的 Amazon GameLift Servers C++ API。具體而言，本文件適用於您使用 `-DBUILD_FOR_UNREAL=1` 選項編譯的程式碼。

適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x

資料類型

- [FProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)

- [FServerParameters](#)
- [FStartMatchBackfillRequest](#)
- [FPlayer](#)
- [FGameLiftDescribePlayerSessionsRequest](#)
- [FStopMatchBackfillRequest](#)
- [FAttributeValue](#)
- [FGameLiftGetFleetRoleCredentialsRequest](#)
- [FGameLiftLongOutcome](#)
- [FGameLiftStringOutcome](#)
- [FGameLiftDescribePlayerSessionsOutcome](#)
- [FGameLiftDescribePlayerSessionsResult](#)
- [FGenericOutcome](#)
- [FGameLiftPlayerSession](#)
- [FGameLiftGetComputeCertificateOutcome](#)
- [FGameLiftGetComputeCertificateResult](#)
- [FGameLiftGetFleetRoleCredentialsOutcome](#)
- [FGetFleetRoleCredentialsResult](#)
- [FGameLiftError](#)
- [列舉](#)

FProcessParameters

此資料類型包含傳送至 Amazon GameLift Servers 的一組參數[ProcessReady\(\)](#)。

屬性	Description
LogParameters	具有在遊戲工作階段期間產生之檔案目錄路徑的物件。會 Amazon GameLift Servers 複製和存放檔案以供日後存取。 Type (類型) : TArray<FString> 必要 : 否

OnHealthCheck

Amazon GameLift Servers 呼叫 從伺服器程序請求運作狀態報告的回呼函數。每 60 秒會 Amazon GameLift Servers 呼叫此函數，並等待 60 秒進行回應。FALSE 如果 TRUE 運作狀態不佳，伺服器程序會傳回。如果未傳回任何回應，會將伺服器程序 Amazon GameLift Servers 記錄為運作狀態不佳。

此屬性是定義為 的委派函數 `DECLARE_DELEGATE_RetVal(bool, FOnHealthCheck)` ；

Type (類型) : FOnHealthCheck

必要：否

OnProcessTerminate

Amazon GameLift Servers 叫用以強制伺服器程序關閉的回呼函數。呼叫此函數後，會 Amazon GameLift Servers 等待 5 分鐘讓伺服器程序關閉，並在關閉伺服器程序之前回應 [ProcessEnding\(\)](#) 呼叫。

Type (類型) : FSimpleDelegate

必要：是

OnStartGameSession

Amazon GameLift Servers 叫用以啟用新遊戲工作階段的回呼函數。會 Amazon GameLift Servers 呼叫此函數以回應用戶端請求 [CreateGameSession](#)。回呼函數會傳遞 [GameSession](#) 物件。

此屬性是定義為的委派函數 `DECLARE_D ELEGATE_OneParam( FOnStartGameSession, Aws::GameLift::Server::Model::GameSession);`

Type (類型) : FOnStartGameSession

必要 : 是

OnUpdateGameSession

Amazon GameLift Servers 叫用以將更新的遊戲工作階段物件傳遞至伺服器程序的回呼函數。會在處理配對回填請求時 Amazon GameLift Servers 呼叫此函數，以提供更新的配對建構器資料。它會傳遞 [GameSession](#) 物件、狀態更新 (updateReason) 和相符回填票證 ID。

此屬性是定義為的委派函數 `DECLARE_D ELEGATE_OneParam( FOnUpdateGameSession, Aws::GameLift::Server::Model::UpdateGameSession);`

Type (類型) : FOnUpdateGameSession

必要 : 否

連線埠

伺服器程序接聽新玩家連線的連接埠號碼。值必須屬於為部署此遊戲伺服器組建之機群所設定的連接埠範圍。此連接埠號碼包含在遊戲工作階段和遊戲工作階段物件中，遊戲工作階段會使用該物件來連接到伺服器程序。

Type (類型) : int

必要 : 是

UpdateGameSession

此資料類型會更新至遊戲工作階段物件，其中包括遊戲工作階段已更新的原因，以及如果回填是用於填入遊戲工作階段中的玩家工作階段，則相關的回填票證 ID。

屬性	Description
GameSession	GameSession 物件。GameSession 物件包含描述遊戲工作階段的屬性。 Type (類型) : Aws::GameLift::Server::GameSession 必要 : 否
UpdateReason	正在更新遊戲工作階段的原因。 Type (類型) : enum class UpdateReason - MATCHMAKING_DATA_UPDATED - BACKFILL_FAILED - BACKFILL_TIMED_OUT - BACKFILL_CANCELLED 必要 : 否
BackfillTicketId	嘗試更新遊戲工作階段的重填票證 ID。

屬性	Description
	Type (類型) : char[] 必要 : 否

GameSession

此資料類型提供遊戲工作階段的詳細資訊。

屬性	Description
GameSessionId	遊戲工作階段的唯一識別符。遊戲工作階段 ARN 的格式如下 : <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> 。 Type (類型) : char[] 必要 : 否
名稱	遊戲工作階段的描述性標籤。 Type (類型) : char[] 必要 : 否
FleetId	遊戲工作階段執行所在的機群的唯一識別符。 Type (類型) : char[] 必要 : 否
MaximumPlayerSessionCount	遊戲工作階段的玩家連線數目上限。 Type (類型) : int 必要 : 否

屬性	Description
連線埠	遊戲工作階段的連接埠號碼。若要連線到 Amazon GameLift Servers 遊戲伺服器，應用程式需要 IP 地址和連接埠號碼。 Type (類型) : int 必要 : 否
IpAddress	遊戲工作階段的 IP 地址。若要連線到 Amazon GameLift Servers 遊戲伺服器，應用程式需要 IP 地址和連接埠號碼。 Type (類型) : char[] 必要 : 否
GameSessionData	一組自訂遊戲工作階段屬性，格式為單一字串值。 Type (類型) : char[] 必要 : 否
MatchmakerData	用於建立遊戲工作階段的配對程序資訊，以 JSON 語法格式為字串。除了使用的配對組態之外，還包含指派給配對的所有玩家的資料，包括玩家屬性和團隊指派。 Type (類型) : char[] 必要 : 否
GameProperties	一組遊戲工作階段的自訂屬性，格式為 key : value 對。這些屬性會與啟動新遊戲工作階段的請求一起傳遞。 Type (類型) : GameProperty[] 必要 : 否

屬性	Description
DnsName	指派給執行遊戲工作階段之執行個體的 DNS 識別符。值的格式如下： - 啟用 TLS 的機群：<code><unique identifier>.<region identifier>.amazon gamelift.com</code>。 - Non-TLS-enabled 的機群：<code>ec2-<unique identifier>.compute.amazonaws.com</code>。 連線至在啟用 TLS 的機群上執行的遊戲工作階段時，您必須使用 DNS 名稱，而非 IP 地址。 Type (類型)：char[] 必要：否

FServerParameters

用來維護 Amazon GameLift ServersAnywhere 伺服器與 Amazon GameLift Servers 服務之間連線的資訊。使用 啟動新的伺服器程序時，會使用此資訊 [InitSDK\(\)](#)。對於託管在 Amazon GameLift Servers 受管 EC2 執行個體上的伺服器，請使用空物件。

屬性	Description
webSocketUrl	當您 RegisterCompute 使用 Amazon GameLift ServersAnywhere 運算資源時，會 <code>GameLiftServerSdkEndpoint</code> Amazon GameLift Servers 傳回。 Type (類型)：char[] 必要：是
processId	向託管遊戲的伺服器程序註冊的唯一識別符。

屬性	Description
hostId	Type (類型) : char[] 必要 : 是 HostID 是當您註冊運算ComputeName 時使用的。如需詳細資訊，請參閱 RegisterCompute。 Type (類型) : char[] 必要 : 是
fleetId	運算註冊目的地機群的唯一識別符。如需詳細資訊，請參閱 RegisterCompute。 Type (類型) : char[] 必要 : 是
authToken	產生的身分驗證字符Amazon GameLift Servers，用於向 驗證您的伺服器Amazon GameLift Servers。如需詳細資訊，請參閱 GetComputeAuthToken。 Type (類型) : char[] 必要 : 是

FStartMatchBackfillRequest

用來建立配對回填請求的資訊。遊戲伺服器會在[StartMatchBackfill\(\)](#)通話Amazon GameLift Servers中將此資訊傳達給。

屬性	Description
GameSessionArn	唯一的遊戲工作階段識別符。API 操作會以 ARN 格式GetGameSessionId 傳回識別符。

屬性	Description
	Type (類型) : char[] 必要 : 是
MatchmakingConfigurationArn	配對建構器用於此請求的唯一識別符，格式為 ARN。原始遊戲工作階段的配對建構器 ARN 位於配對建構器資料屬性中的遊戲工作階段物件中。進一步了解使用配對建構器資料中的配對建構器資料。 Type (類型) : char[] 必要 : 是
Players	一組資料，代表遊戲工作階段中的所有玩家。配對建構器使用此項資訊搜尋適合配對現有玩家的新玩家。 Type (類型) : TArray<FPlayer> 必要 : 是
TicketId	配對或配對回填請求票證的唯一識別符。如果您不提供值，Amazon GameLift Servers 會產生值。您可使用此識別項依據需求追蹤配對回填票證狀態或取消要求。 Type (類型) : char[] 必要 : 否

FPlayer

此資料類型代表配對中的玩家。開始配對請求時，玩家會有玩家 ID、屬性和可能的延遲資料。會在配對完成後 Amazon GameLift Servers 新增隊伍資訊。

屬性	Description
LatencyInMS	一組以毫秒為單位的值，指出玩家連線到位置時所經歷的延遲量。 如果使用此屬性，則玩家只會符合列出的位置。若配對構建器有評估玩家延遲的規則，玩家則必須回報延遲度，方可配對。 Type (類型) : TMap>FString, int32< 必要 : 否
PlayerAttributes	包含玩家資訊的金鑰：值對集合，可用於配對。玩家屬性金鑰必須符合配對規則集中使用的 PlayerAttributes。 如需玩家屬性的詳細資訊，請參閱 Attribute Value。 Type (類型) : TMap>FString, FAttributeValue< 必要 : 否
PlayerId	玩家的唯一識別符。 Type (類型) : std::string 必要 : 否
團隊	玩家在配對中獲指派的隊伍名稱。您可以在配對規則集中定義團隊名稱。 Type (類型) : FString 必要 : 否

FGameLiftDescribePlayerSessionsRequest

指定要擷取哪些玩家工作階段的物件。伺服器程序會提供此資訊給 [DescribePlayerSessions\(\)](#)。

Amazon GameLift Servers

屬性	Description
GameSessionId	唯一的遊戲工作階段識別符。請使用此參數要求特定遊戲工作階段的所有玩家工作階段。 遊戲工作階段 ID 格式為 FString。GameSessionID 是自訂 ID 字串或 Type (類型) : std::string 必要 : 否
PlayerSessionId	玩家工作階段的唯一識別符。使用此參數請求單一特定玩家工作階段。 Type (類型) : FString 必要 : 否
PlayerId	玩家的唯一識別符。使用此參數來請求特定玩家的所有玩家工作階段。請參閱 產生玩家 IDs。 Type (類型) : FString 必要 : 否
PlayerSessionStatusFilter	要篩選結果的玩家工作階段狀態。可能的玩家工作階段狀態包括： • RESERVED – 已收到玩家工作階段請求，但玩家尚未連線到伺服器程序或已經過驗證。 • ACTIVE – 播放器已由伺服器程序驗證並連線。 • 已完成 – 玩家連線中斷。

屬性	Description
	TIMEDOUT – 已收到玩家工作階段請求，但玩家未連線或未在逾時限制 (60 秒) 內驗證。 Type (類型) : FString 必要 : 否
NextToken	表示結果下一頁開始的字符。若要指定結果集的開頭，請不要提供值。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : FString 必要 : 否
限制	回傳結果的數量上限。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : int 必要 : 否

FStopMatchBackfillRequest

用來取消配對回填請求的資訊。遊戲伺服器會在[StopMatchBackfill\(\)](#)通話中將此資訊傳達給 Amazon GameLift Servers 服務。

屬性	Description
GameSessionArn	取消請求的唯一遊戲工作階段識別符。 Type (類型) : FString 必要 : 是
MatchmakingConfigurationArn	此請求傳送至之配對建構器的唯一識別符。

屬性	Description
	Type (類型) : FString
	必要 : 是
TicketId	要取消之回填請求票證的唯一識別符。
	Type (類型) : FString
	必要 : 是

FAttributeValue

在 [FPlayer](#) 屬性鍵/值對中使用這些值。此物件可讓您使用任何有效的資料類型來指定屬性值：字串、數字、字串陣列或資料映射。每個 `AttributeValue` 物件只能使用其中一個可用的屬性。

屬性	描述
attrType	指定屬性值的類型。 類型 : <code>FAttributeType</code> 列舉 值。 必要 : 否
S	代表字串屬性值。 Type (類型) : <code>FString</code> 必要 : 否
N	代表數值屬性值。 Type (類型) : <code>double</code> 必要 : 否
SL	代表字串屬性值的陣列。 Type (類型) : <code>TArray<FString></code>

屬性	描述
	必要：否
SDM	代表字串索引鍵和雙值的字典。 Type (類型) : TMap<FString, double> 必要：否

FGameLiftGetFleetRoleCredentialsRequest

此資料類型提供角色登入資料，可將 資源 AWS 的有限存取權延伸到遊戲伺服器。如需詳細資訊，請參閱 [設定的 IAM 服務角色 Amazon GameLift Servers](#)。

屬性	Description
RoleArn	服務角色的 Amazon Resource Name (ARN) ， 可延伸 AWS 資源的有限存取權。 Type (類型) : FString 必要：否
RoleSessionName	描述使用角色登入資料的工作階段名稱。 Type (類型) : FString 必要：否

FGameLiftLongOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : long

屬性	描述
	必要：否
ResultWithOwnership	動作的結果會轉換為 rvalue，以便呼叫程式碼可以取得物件的擁有權。 Type (類型)：long&& 必要：否
成功	動作是否成功。 Type (類型)：bool 必要：是
錯誤	動作失敗時發生的錯誤。 Type (類型)：the section called “FGameLiftError” 必要：否

FGameLiftStringOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型)：FString 必要：否
ResultWithOwnership	動作的結果會轉換為 rvalue，以便呼叫程式碼可以取得物件的擁有權。 Type (類型)：FString&& 必要：否

屬性	描述
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “FGameLiftError” 必要 : 否

FGameLiftDescribePlayerSessionsOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “FGameLiftDescribePlayerSessionsResult” 必要 : 否
ResultWithOwnership	動作的結果會轉換為 rvalue，以便呼叫程式碼可以取得物件的擁有權。 Type (類型) : FGameLiftDescribePlayerSessionsResult&& 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是

屬性	描述
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “FGameLiftError” 必要 : 否

FGameLiftDescribePlayerSessionsResult

屬性	描述
PlayerSessions	Type (類型) : TArray<FGameLiftPlayerSession> 必要 : 是
NextToken	表示結果下一頁開始的字符。若要指定結果集的開頭，請不要提供值。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型) : FString 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “FGameLiftError” 必要 : 否

FGenericOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “FGameLiftError” 必要 : 否

FGameLiftPlayerSession

屬性	描述
CreationTime	Type (類型) : long 必要 : 是
FleetId	Type (類型) : FString 必要 : 是
GameSessionId	Type (類型) : FString 必要 : 是
IpAddress	Type (類型) : FString 必要 : 是
PlayerData	Type (類型) : FString 必要 : 是
PlayerId	Type (類型) : FString

屬性	描述
	必要：是
PlayerSessionId	Type (類型) : FString 必要：是
連線埠	Type (類型) : int 必要：是
Status	類型 : PlayerSessionStatus 列舉 。 必要：是
TerminationTime	Type (類型) : long 必要：是
DnsName	Type (類型) : FString 必要：是

FGameLiftGetComputeCertificateOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。 Type (類型) : the section called “FGameLiftGetComputeCertificateResult” 必要：否
ResultWithOwnership	動作的結果會轉換為 rvalue，以便呼叫程式碼可以取得物件的擁有權。

屬性	描述
	Type (類型) : FGameLiftGetComputeCertificateResult&& 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “FGameLiftError” 必要 : 否

FGameLiftGetComputeCertificateResult

運算上 TLS 憑證的路徑和運算的主機名稱。

屬性	描述
CertificatePath	Type (類型) : FString 必要 : 是
ComputeName	Type (類型) : FString 必要 : 是

FGameLiftGetFleetRoleCredentialsOutcome

此資料類型來自 動作，並產生具有下列屬性的物件：

屬性	描述
結果	動作的結果。

屬性	描述
	Type (類型) : the section called “FGetFleetRoleCredentialsResult” 必要 : 否
ResultWithOwnership	動作的結果會轉換為 rvalue，以便呼叫程式碼可以取得物件的擁有權。 Type (類型) : FGameLiftGetFleetRoleCredentialsResult&& 必要 : 否
成功	動作是否成功。 Type (類型) : bool 必要 : 是
錯誤	動作失敗時發生的錯誤。 Type (類型) : the section called “FGameLiftError” 必要 : 否

FGetFleetRoleCredentialsResult

屬性	描述
AccessKeyId	要驗證並提供 AWS 資源存取權的存取金鑰 ID。 Type (類型) : FString 必要 : 否
AssumedRoleId	服務角色所屬的使用者 ID。 Type (類型) : FString

屬性	描述
	必要：否
AssumedRoleUserArn	服務角色所屬使用者的 Amazon Resource Name (ARN)。 Type (類型)：FString 必要：否
過期	工作階段登入資料過期前的時間長度。 Type (類型)：FDateTime 必要：否
SecretAccessKey	用於身分驗證的私密存取金鑰 ID。 Type (類型)：FString 必要：否
SessionToken	識別目前作用中工作階段與 AWS 資源互動的字符。 Type (類型)：FString 必要：否
成功	動作是否成功。 Type (類型)：bool 必要：是
錯誤	動作失敗時發生的錯誤。 Type (類型)： the section called “GameLiftError” 必要：否

FGameLiftError

屬性	描述
ErrorType	錯誤類型。 類型：GameLiftErrorType 列舉 。 必要：否
ErrorMessage	錯誤的名稱。 Type (類型)：std::string 必要：否
ErrorMessage	錯誤訊息。 Type (類型)：std::string 必要：否

列舉

為適用於 Amazon GameLift Servers (非真實) 的伺服器 SDK 定義的列舉定義如下：

FAttributeType

- NONE
- STRING
- DOUBLE
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

指出錯誤類型的字串值。有效值包含：

- SERVICE_CALL_FAILED – 呼叫 AWS 服務失敗。
- LOCAL_CONNECTION_FAILED – 與的本機連線Amazon GameLift Servers失敗。

- NETWORK_NOT_INITIALIZED – 網路尚未初始化。
- GAMESESSION_ID_NOT_SET – 尚未設定遊戲工作階段 ID。
- BAD_REQUEST_EXCEPTION
- INTERNAL_SERVICE_EXCEPTION
- ALREADY_INITIALIZED – Amazon GameLift Servers 伺服器或用戶端已使用 Initialize() 初始化。
- FLEET_MISMATCH – 目標機群與 gameSession 或 playerSession 的機群不相符。
- GAMELIFT_CLIENT_NOT_INITIALIZED – Amazon GameLift Servers 用戶端尚未初始化。
- GAMELIFT_SERVER_NOT_INITIALIZED – Amazon GameLift Servers 伺服器尚未初始化。
- GAME_SESSION_ENDED_FAILED – 的 伺服器開發套件Amazon GameLift Servers無法聯絡 服務來報告遊戲工作階段已結束。
- GAME_SESSION_NOT_READY – Amazon GameLift Servers 伺服器遊戲工作階段未啟用。
- GAME_SESSION_READY_FAILED – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務來報告遊戲工作階段已就緒。
- INITIALIZATION_MISMATCH – 在 Server : : Initialize() 之後呼叫用戶端方法，反之亦然。
- NOT_INITIALIZED – Amazon GameLift Servers 伺服器或用戶端尚未以 Initialize() 初始化。
- NO_TARGET_ALIASID_SET – 尚未設定目標aliasId。
- NO_TARGET_FLEET_SET – 尚未設定目標機群。
- PROCESS_ENDING_FAILED – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務來報告程序即將結束。
- PROCESS_NOT_ACTIVE – 伺服器程序尚未啟動、未繫結至 GameSession，且無法接受或處理 PlayerSessions。
- PROCESS_NOT_READY – 伺服器程序尚未準備好啟用。
- PROCESS_READY_FAILED – 的伺服器 SDK Amazon GameLift Servers 無法聯絡 服務以回報程序已就緒。
- SDK_VERSION_DETECTION_FAILED – SDK 版本偵測失敗。
- STX_CALL_FAILED – 呼叫 XStx 伺服器後端元件失敗。
- STX_INITIALIZATION_FAILED – XStx 伺服器後端元件無法初始化。
- UNEXPECTED_PLAYER_SESSION – 伺服器遇到未註冊的玩家工作階段。
- WEBSOCKET_CONNECT_FAILURE

- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE – 將訊息傳送至 GameLift Service WebSocket 的可重試失敗。
- WEBSOCKET_SEND_MESSAGE_FAILURE – 無法傳送訊息至 GameLift Service WebSocket。
- MATCH_BACKFILL_REQUEST_VALIDATION – 驗證請求失敗。
- PLAYER_SESSION_REQUEST_VALIDATION – 驗證請求失敗。

EPlayerSessionCreationPolicy

字串值代表遊戲工作階段是否可接受新玩家。有效值包含：

- ACCEPT_ALL – 接受所有新玩家工作階段。
- DENY_ALL – 拒絕所有新玩家工作階段。
- NOT_SET – 遊戲工作階段未設定為接受或拒絕新的玩家工作階段。

EPlayerSessionStatus

- ACTIVE
- COMPLETED (已完成)
- NOT_SET
- RESERVED
- 逾時

[適用於 -- 資料類型的 C++ \(非真實\) 伺服器 SDK Amazon GameLift Servers 5.x](#)

主題

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)

- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Destroy\(\)](#)

GetSdkVersion()

傳回內建至伺服器程序的目前開發套件版本編號。

語法

```
FGameLiftStringOutcome GetSdkVersion();
```

傳回值

如果成功，將目前開發套件版本以 [the section called “FGameLiftStringOutcome”](#) 物件傳回。傳回的物件包含版本編號（範例 5.0.0）。如果不成功，則會傳回錯誤訊息。

範例

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

初始化受管 EC2 機群的 Amazon GameLift Servers SDK。在啟動時，先呼叫此方法，再 Amazon GameLift Servers 進行任何其他與相關的初始化。此方法會從主機環境讀取伺服器參數，以設定伺服

器與服務之間的通訊Amazon GameLift Servers。它使用等冪符記，因此您可以在失敗時安全地重試此呼叫。

語法

```
FGameLiftGenericOutcome InitSDK()
```

傳回值

如果成功，會傳回 `InitSdkOutcome` 物件，指出伺服器程序已準備好呼叫 [ProcessReady\(\)](#)。

範例

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
FGameLiftGenericOutcome initSdkOutcome = GameLiftSdkModule->InitSDK();
```

InitSDK()

初始化 Anywhere 機群或受管容器機群的 Amazon GameLift Servers SDK。在啟動時，先呼叫此方法，再Amazon GameLift Servers進行任何其他與 相關的初始化。此方法需要明確的伺服器參數，才能設定伺服器和服務之間的通訊Amazon GameLift Servers。它使用等冪符記，因此您可以在失敗時安全地重試此呼叫。

語法

```
FGameLiftGenericOutcome InitSDK(serverParameters)
```

參數

[FServerParameters](#)

若要在 Amazon GameLift ServersAnywhere 機群上初始化遊戲伺服器，請使用下列資訊建構 `ServerParameters` 物件：

- 用來連線至遊戲伺服器的 WebSocket URL。
- 用於託管遊戲伺服器的程序 ID。
- 託管遊戲伺服器程序的運算 ID。
- 包含 Amazon GameLift ServersAnywhere 運算的Amazon GameLift Servers機群 ID。
- Amazon GameLift Servers 操作產生的授權字符。

傳回值

如果成功，會傳回 `InitSdkOutcome` 物件，指出伺服器程序已準備好呼叫 [ProcessReady\(\)](#)。

Note

如果部署到 Anywhere 機群的遊戲組建呼叫 `InitSDK()` 失敗，請檢查建立組建資源時使用的 `ServerSdkVersion` 參數。您必須將此值明確設定為正在使用的伺服器 SDK 版本。此參數的預設值為 4.x，不相容。若要解決此問題，請建立新的組建，並將其部署到新的機群。

範例

```
//Define the server parameters
FServerParameters serverParameters;
parameters.m_authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
parameters.m_fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
parameters.m_hostId = "HardwareAnywhere";
parameters.m_processId = "PID1234";
parameters.m_webSocketUrl = "wss://us-west-1.api.amazongamelift.com";

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
FGameLiftGenericOutcome initSdkOutcome = GameLiftSdkModule->InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift Servers 通知伺服器程序已準備好託管遊戲工作階段。叫用 後呼叫此方法 [InitSDK\(\)](#)。每個程序只能呼叫此方法一次。

語法

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

參數

`processParameters`

傳達下列伺服器程序相關資訊的 [FProcessParameters](#) 物件：

- 在遊戲伺服器程式碼中實作的回呼方法名稱，Amazon GameLift Servers服務會叫用它來與伺服器程序通訊。
- 伺服器程序正在接聽的埠號。
- 任何要 Amazon GameLift Servers 擷取並存放的遊戲工作階段特定檔案路徑。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會說明 [ProcessReady\(\)](#) 呼叫和委派函數的實作。

```
//Calling ProcessReady tells Amazon GameLift Servers this game server is ready to
receive incoming game sessions!
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready"));
FGameLiftGenericOutcome processReadyOutcome = GameLiftSdkModule-
>ProcessReady(*params);
```

ProcessEnding()

Amazon GameLift Servers 通知伺服器程序正在終止。在所有其他清除任務（包括關閉作用中的遊戲工作階段）之後，以及在終止程序之前，呼叫此方法。根據的結果ProcessEnding()，程序會以成功 (0) 或錯誤 (-1) 結束並產生機群事件。如果程序因錯誤而終止，產生的機群事件為SERVER_PROCESS_TERMINATED_UNHEALTHY)。

語法

```
FGameLiftGenericOutcome ProcessEnding()
```

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
//OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback before
shutting down an instance hosting this game server.
```

```
//It gives this game server a chance to save its state, communicate with services,  
etc., before being shut down.  
//In this case, we simply tell Amazon GameLift Servers we are indeed going to shutdown.  
params->OnTerminate.BindLambda(=[]() {  
    UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));  
    GameLiftSdkModule->ProcessEnding();  
});
```

ActivateGameSession()

Amazon GameLift Servers 通知伺服器程序已啟用遊戲工作階段，現在已準備好接收玩家連線。在所有遊戲工作階段初始化之後，應該呼叫此動作做為回onStartGameSession()呼函數的一部分。

語法

```
FGameLiftGenericOutcome ActivateGameSession()
```

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例顯示ActivateGameSession()稱為onStartGameSession()委派函數的一部分。

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

UpdatePlayerSessionCreationPolicy()

更新目前遊戲工作階段的能力，以接受新的玩家工作階段。遊戲工作階段可設定為接受或拒絕所有新的玩家工作階段。

語法

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy policy)
```

參數

playerCreationSessionPolicy

字串值代表遊戲工作階段是否可接受新玩家。

有效值包含：

- ACCEPT_ALL – 接受所有新玩家工作階段。
- DENY_ALL – 拒絕所有新玩家工作階段。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例設定目前遊戲工作階段的加入政策為可接受所有玩家。

```
FGameLiftGenericOutcome outcome = GameLiftSdkModule->UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::EPlayerSessionCreationPolicy::ACCEPT_A
```

GetGameSessionId()

擷取作用中伺服器程序託管的遊戲工作階段 ID。

對於未使用遊戲工作階段啟用的閒置程序，呼叫會傳回 [the section called “FGameLiftError”](#)。

語法

```
FGameLiftStringOutcome GetGameSessionId()
```

參數

此動作沒有參數。

傳回值

如果成功，則會把遊戲工作階段 ID 當成 [the section called "FGameLiftStringOutcome"](#) 物件傳回。如果不成功，會傳回錯誤訊息。」

對於未使用遊戲工作階段啟用的閒置程序，呼叫會傳回 `Success=True` 和 `GameSessionId=""`。

範例

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

GetTerminationTime()

若設有終止時間，即傳回伺服器程序排定關閉的時間。伺服器程序在收到來自的 `onProcessTerminate()` 呼後會採取動作 Amazon GameLift Servers。Amazon GameLift `Server::onProcessTerminate()` 會呼叫，原因如下：

- 當伺服器程序回報運作狀態不佳或未回應時 Amazon GameLift Servers。
- 在縮減規模事件期間終止執行個體時。
- 當執行個體因 [spot-instance 中斷](#) 而終止時。

語法

```
AwsDateTimeOutcome GetTerminationTime()
```

傳回值

如果成功，會將終止時間傳回為 `AwsDateTimeOutcome` 物件。此值為終止時間，以自以來經過的刻度表示 `0001 00:00:00`。例如，日期時間值 `2020-09-13 12:26:40 -0000Z` 等於 `6373559680000000000` 刻度。如果沒有可用的終止時間，會傳回錯誤訊息。

如果程序尚未收到 `ProcessParameters.OnProcessTerminate()` 回呼，則會傳回錯誤訊息。如需關閉伺服器程序的詳細資訊，請參閱 [回應伺服器程序關閉通知](#)。

範例

```
AwsDateTimeOutcome TermTimeOutcome = GameLiftSdkModule->GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift Servers 通知具有指定玩家工作階段 ID 的玩家已連線至伺服器程序，且需要驗證。Amazon GameLift Servers 會驗證玩家工作階段 ID 是否有效。驗證玩家工作階段後，會將玩家位置 Amazon GameLift Servers 的狀態從 RESERVED 變更為 ACTIVE。

語法

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

參數

playerSessionId

建立新玩家工作階段 Amazon GameLift Servers 時由 發出的唯一 ID。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會處理連線請求，其中包含驗證和拒絕非有效的玩家工作階段 IDs。

```
bool GameLiftManager::AcceptPlayerSession(const FString& playerId, const
FString& gameId)
{
    #if WITH_GAMELIFT
    UE_LOG(GameServerLog, Log, TEXT("Accepting GameLift PlayerSession: %s . PlayerId:
%s"), *playerSessionId, *playerId);
    FString gsId = GetCurrentGameSessionId();
    if (gsId.IsEmpty()) {
        UE_LOG(GameServerLog, Log, TEXT("No GameLift GameSessionId. Returning early!"));
        return false;
    }
}
```

```
if (!GameLiftSdkModule->AcceptPlayerSession(playerSessionId).IsSuccess()) {
    UE_LOG(GameServerLog, Log, TEXT("PlayerSession not Accepted.));
    return false;
}

// Add PlayerSession from internal data structures keeping track of connected players
connectedPlayerSessionIds.Add(playerSessionId);
idToPlayerSessionMap.Add(playerSessionId, PlayerSession{ playerId,
playerSessionId });
return true;
#else
return false;
#endif
}
```

RemovePlayerSession()

Amazon GameLift Servers 通知玩家已中斷與伺服器程序的連線。作為回應，Amazon GameLift Servers 會將玩家位置變更為可用。

語法

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

參數

playerSessionId

建立新玩家工作階段Amazon GameLift Servers時由 發出的唯一 ID。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
bool GameLiftManager::RemovePlayerSession(const FString& playerSessionId)
{
    #if WITH_GAMELIFT
    UE_LOG(GameServerLog, Log, TEXT("Removing GameLift PlayerSession: %s"),
*playerSessionId);
```

```
if (!GameLiftSdkModule->RemovePlayerSession(playerSessionId).IsSuccess()) {
    UE_LOG(GameServerLog, Log, TEXT("PlayerSession Removal Failed"));
    return false;
}

// Remove PlayerSession from internal data structures that are keeping track of
connected players
connectedPlayerSessionIds.Remove(playerSessionId);
idToPlayerSessionMap.Remove(playerSessionId);

// end the session if there are no more players connected
if (connectedPlayerSessionIds.Num() == 0) {
    EndSession();
}

return true;
#else
return false;
#endif
}
```

DescribePlayerSessions()

擷取玩家工作階段資料，其中包括設定、工作階段中繼資料和玩家資料。使用此方法取得下列相關資訊：

- 單一玩家工作階段
- 遊戲工作階段中的所有玩家工作階段
- 與單一玩家 ID 相關聯的所有玩家工作階段

語法

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const
FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

參數

[FGameLiftDescribePlayerSessionsRequest](#)

描述要擷取哪些玩家工作階段的 [the section called “FGameLiftDescribePlayerSessionsRequest”](#) 物件。

傳回值

如果成功，會傳回 [the section called “FGameLiftDescribePlayerSessionsOutcome”](#) 物件，內含一組與請求參數相符的玩家工作階段物件。

範例

此範例會請求主動連線到指定遊戲工作階段的所有玩家工作階段。透過省略 NextToken 並將限制值設定為 10，會Amazon GameLift Servers傳回符合請求的前 10 個玩家工作階段記錄。

```
void GameLiftManager::DescribePlayerSessions()
{
    #if WITH_GAMELIFT
    FString localPlayerSessions;
    for (auto& psId : connectedPlayerSessionIds)
    {
        PlayerSession ps = idToPlayerSessionMap[psId];
        localPlayerSessions += FString::Printf(TEXT("%s : %s ; "), *(ps.playerSessionId),
*(ps.playerId));
    }
    UE_LOG(GameServerLog, Log, TEXT("LocalPlayerSessions: %s"), *localPlayerSessions);

    UE_LOG(GameServerLog, Log, TEXT("Describing PlayerSessions in this GameSession"));
    FGameLiftDescribePlayerSessionsRequest request;
    request.m_gameSessionId = GetCurrentGameSessionId();

    FGameLiftDescribePlayerSessionsOutcome outcome = GameLiftSdkModule-
>DescribePlayerSessions(request);
    LogDescribePlayerSessionsOutcome(outcome);
    #endif
}
```

StartMatchBackfill()

此動作會傳送請求，以便替 FlexMatch 所建立的遊戲工作階段開放空位找到新玩家。如需詳細資訊，請參閱[FlexMatch回填功能](#)。

此為非同步動作。如果新玩家相符，會使用回呼函數 Amazon GameLift Servers交付更新的配對建構器資料OnUpdateGameSession()。

一個伺服器程序一次僅能有一個使用中的配對回填請求。若要發送新請求，請先呼叫[StopMatchBackfill\(\)](#) 取消原始請求。

語法

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest
&startBackfillRequest);
```

參數

[FStartMatchBackfillRequest](#)

StartMatchBackfillRequest 物件，可傳達下列資訊：

- 指派給回填請求的票證 ID。此資訊是選用的；如果未提供 ID，Amazon GameLift Servers 將產生一個 ID。
- 傳送請求對象的配對建構器。必須填入完整的組態 ARN。此值位於遊戲工作階段的配對建構器資料中。
- 要回填的遊戲工作階段 ID。
- 遊戲工作階段目前玩家的可用配對資料。

傳回值

傳回具有相符回填票證 ID 或失敗並顯示錯誤訊息的 StartMatchBackfillOutcome 物件。

範例

```
FGameLiftStringOutcome FGameLiftServerSDKModule::StartMatchBackfill(const
FStartMatchBackfillRequest& request)
{
    #if WITH_GAMELIFT
    Aws::GameLift::Server::Model::StartMatchBackfillRequest sdkRequest;
    sdkRequest.SetTicketId(TCHAR_TO_UTF8(*request.m_ticketId));
    sdkRequest.SetGameSessionArn(TCHAR_TO_UTF8(*request.m_gameSessionArn));

    sdkRequest.SetMatchmakingConfigurationArn(TCHAR_TO_UTF8(*request.m_matchmakingConfigurationArn));
    for (auto player : request.m_players) {
        Aws::GameLift::Server::Model::Player sdkPlayer;
        sdkPlayer.SetPlayerId(TCHAR_TO_UTF8(*player.m_playerId));
        sdkPlayer.SetTeam(TCHAR_TO_UTF8(*player.m_team));
        for (auto entry : player.m_latencyInMs) {
            sdkPlayer.WithLatencyMs(TCHAR_TO_UTF8(*entry.Key), entry.Value);
        }
    }
}
```

```

    std::map<std::string, Aws::GameLift::Server::Model::AttributeValue>
    sdkAttributeMap;
    for (auto attributeEntry : player.m_playerAttributes) {
        FAttributeValue value = attributeEntry.Value;
        Aws::GameLift::Server::Model::AttributeValue attribute;
        switch (value.m_type) {
            case FAttributeType::STRING:
                attribute =
                Aws::GameLift::Server::Model::AttributeValue(TCHAR_TO_UTF8(*value.m_S));
                break;
            case FAttributeType::DOUBLE:
                attribute = Aws::GameLift::Server::Model::AttributeValue(value.m_N);
                break;
            case FAttributeType::STRING_LIST:
                attribute =
                Aws::GameLift::Server::Model::AttributeValue::ConstructStringList();
                for (auto sl : value.m_SL) {
                    attribute.AddString(TCHAR_TO_UTF8(*sl));
                };
                break;
            case FAttributeType::STRING_DOUBLE_MAP:
                attribute =
                Aws::GameLift::Server::Model::AttributeValue::ConstructStringDoubleMap();
                for (auto sdm : value.m_SDM) {
                    attribute.AddStringAndDouble(TCHAR_TO_UTF8(*sdm.Key), sdm.Value);
                };
                break;
        }
        sdkPlayer.WithPlayerAttribute((TCHAR_TO_UTF8(*attributeEntry.Key)), attribute);
    }
    sdkRequest.AddPlayer(sdkPlayer);
}
auto outcome = Aws::GameLift::Server::StartMatchBackfill(sdkRequest);
if (outcome.IsSuccess()) {
    return FGameLiftStringOutcome(outcome.GetResult().GetTicketId());
}
else {
    return FGameLiftStringOutcome(FGameLiftError(outcome.GetError()));
}
#else
return FGameLiftStringOutcome("");
#endif
}

```

StopMatchBackfill()

取消作用中的配對回填請求。如需詳細資訊，請參閱[FlexMatch回填功能](#)。

語法

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
&stopBackfillRequest);
```

參數

[FStopMatchBackfillRequest](#)

識別要取消之配對票證的 StopMatchBackfillRequest 物件：

- 指派給回填請求的票證 ID。
- 回填請求傳送到的配對建構器。
- 與回填請求相關聯的遊戲工作階段。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
FGameLiftGenericOutcome FGameLiftServerSDKModule::StopMatchBackfill(const
FStopMatchBackfillRequest& request)
{
    #if WITH_GAMELIFT
    Aws::GameLift::Server::Model::StopMatchBackfillRequest sdkRequest;
    sdkRequest.SetTicketId(TCHAR_TO_UTF8(*request.m_ticketId));
    sdkRequest.SetGameSessionArn(TCHAR_TO_UTF8(*request.m_gameSessionArn));

    sdkRequest.SetMatchmakingConfigurationArn(TCHAR_TO_UTF8(*request.m_matchmakingConfigurationArn));
    auto outcome = Aws::GameLift::Server::StopMatchBackfill(sdkRequest);
    if (outcome.IsSuccess()) {
        return FGameLiftGenericOutcome(nullptr);
    }
    else {
        return FGameLiftGenericOutcome(FGameLiftError(outcome.GetError()));
    }
    #else
```

```
return FGameLiftGenericOutcome(nullptr);
#endif
}
```

GetComputeCertificate()

擷取用於加密 Amazon GameLift ServersAnywhere 運算資源與 之間網路連線的 TLS 憑證路徑 Amazon GameLift Servers。您可以在向 Amazon GameLift ServersAnywhere 機群註冊運算裝置時使用憑證路徑。如需詳細資訊，請參閱 [RegisterCompute](#)。

語法

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
```

傳回值

傳回包含下列項目的GetComputeCertificateResponse物件：

- CertificatePath：運算資源上 TLS 憑證的路徑。
- HostName：運算資源的主機名稱。

範例

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
{
    #if WITH_GAMELIFT
    auto outcome = Aws::GameLift::Server::GetComputeCertificate();
    if (outcome.IsSuccess()) {
        auto& outres = outcome.GetResult();
        FGameLiftGetComputeCertificateResult result;
        result.m_certificate_path = UTF8_T0_TCHAR(outres.GetCertificatePath());
        result.m_computeName = UTF8_T0_TCHAR(outres.GetComputeName());
        return FGameLiftGetComputeCertificateOutcome(result);
    }
    else {
        return FGameLiftGetComputeCertificateOutcome(FGameLiftError(outcome.GetError()));
    }
    #else
    return FGameLiftGetComputeCertificateOutcome(FGameLiftGetComputeCertificateResult());
    #endif
}
```


GetFleetRoleCredentials()

擷取授權 與其他 互動的 IAM 角色登入Amazon GameLift Servers資料 AWS 服務。如需詳細資訊，請參閱[與機群中的其他 AWS 資源通訊](#)。

語法

```
FGameLiftGetFleetRoleCredentialsOutcome
FGameLiftServerSDKModule::GetFleetRoleCredentials(const
FGameLiftGetFleetRoleCredentialsRequest &request)
```

參數

[FGameLiftGetFleetRoleCredentialsRequest](#)

傳回值

其會傳回 [the section called “FGameLiftGetFleetRoleCredentialsOutcome”](#) 物件。

範例

```
FGameLiftGetFleetRoleCredentialsOutcome
FGameLiftServerSDKModule::GetFleetRoleCredentials(const
FGameLiftGetFleetRoleCredentialsRequest &request)
{
    #if WITH_GAMELIFT
    Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest sdkRequest;
    sdkRequest.SetRoleArn(TCHAR_TO_UTF8(*request.m_roleArn));
    sdkRequest.SetRoleSessionName(TCHAR_TO_UTF8(*request.m_roleSessionName));

    auto outcome = Aws::GameLift::Server::GetFleetRoleCredentials(sdkRequest);

    if (outcome.IsSuccess()) {
        auto& outres = outcome.GetResult();
        FGameLiftGetFleetRoleCredentialsResult result;
        result.m_assumedUserRoleArn = UTF8_TO_TCHAR(outres.GetAssumedUserRoleArn());
        result.m_assumedRoleId = UTF8_TO_TCHAR(outres.GetAssumedRoleId());
        result.m_accessKeyId = UTF8_TO_TCHAR(outres.GetAccessKeyId());
        result.m_secretAccessKey = UTF8_TO_TCHAR(outres.GetSecretAccessKey());
        result.m_sessionToken = UTF8_TO_TCHAR(outres.GetSessionToken());
        result.m_expiration = FDateTime::FromUnixTimestamp(outres.GetExpiration());
        return FGameLiftGetFleetRoleCredentialsOutcome(result);
    }
    else {
```

```
    return FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftError(outcome.GetError()));
}
#else
return
FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftGetFleetRoleCredentialsResult());
#endif
}
```

Destroy()

從記憶體釋放Amazon GameLift Servers遊戲伺服器 SDK。最佳實務是在終止程序之後ProcessEnding()和之前呼叫此方法。如果您使用的是 Anywhere 機群，而且不是在每次遊戲工作階段後終止伺服器程序，請呼叫 Destroy()，然後在通知 Amazon GameLift Servers 程序已準備好使用託管遊戲工作階段之前InitSDK()重新初始化 ProcessReady()。

語法

```
FGameLiftGenericOutcome Destroy();
```

參數

沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
// First call ProcessEnding()
FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule->ProcessEnding();

// Then call Destroy() to free the SDK from memory
FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();

// Exit the process with success or failure
if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess())
{
    UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
}
else {
    if (!processEndingOutcome.IsSuccess()) {
        const FGameLiftError& error = processEndingOutcome.GetError();
```

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

第 4 Amazon GameLift Servers 版及更早版本的伺服器 SDK

此參考記錄適用於 Amazon GameLift Servers、4.x 版及更早版本的伺服器 SDK。伺服器開發套件提供核心功能，您的遊戲伺服器可用來與服務通訊 Amazon GameLift Servers。例如，您的遊戲伺服器會收到來自服務的提示，以啟動和停止遊戲工作階段，並定期為服務提供遊戲工作階段狀態更新。將遊戲伺服器與伺服器 SDK 整合後，再部署它們進行託管。

使用此伺服器 SDK 參考整合您的自訂多玩家遊戲伺服器，以便與進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

適用於的伺服器 SDK 的最新主要版本 Amazon GameLift Servers 為 5.x。下列託管功能需要更新至 5.x 版：

- Amazon GameLift Servers 隨處
- Amazon GameLift Servers 受管容器
- Amazon GameLift Servers Unreal Engine 和 Unity 的外掛程式

主題

- [適用於 Amazon GameLift Servers 4.x 的 C++ 伺服器 SDK -- 動作](#)
- [適用於 Amazon GameLift Servers 4.x 的 C# 伺服器 SDK -- 動作](#)
- [適用於 -- Amazon GameLift Servers 動作的伺服器 SDK \(Unreal\)](#)

適用於 Amazon GameLift Servers 4.x 的 C++ 伺服器 SDK -- 動作

使用伺服器 SDK 參考來整合多玩家遊戲，以便與進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

Note

此參考適用於 的舊版伺服器 SDK Amazon GameLift Servers。如需最新版本，請參閱[適用於 -- 動作的 C++ 伺服器 SDK Amazon GameLift Servers 5.x](#)。

適用於 Amazon GameLift Servers 4.x 的 C++ 伺服器 SDK -- 資料類型

使用伺服器 SDK 參考整合多玩家遊戲，以便與 進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

Note

此參考適用於舊版 的伺服器 SDK Amazon GameLift Servers。如需最新版本，請參閱[適用於 -- 資料類型的 C++ 伺服器 SDK Amazon GameLift Servers 5.x](#)。

此 API 定義於 `GameLiftServerAPI.h`、`LogParameters.h`、`ProcessParameters.h`。

[適用於 Amazon GameLift Servers 4.x 的 C++ 伺服器 SDK -- 動作](#)

`DescribePlayerSessionsRequest`

此資料類型用於指定要擷取的玩家工作階段，您可利用下列方式使用：

- 提供 `PlayerSessionId` 以要求特定玩家工作階段。
- 提供 `GameSessionId` 以要求特定遊戲工作階段之中的所有玩家工作階段。
- 提供 `PlayerId` 以要求特定玩家的所有玩家工作階段。

對於大量的玩家工作階段，可使用分頁參數於循序區塊擷取結果。

目錄

`GameSessionId`

獨一無二的遊戲工作階段識別項。請使用此參數要求特定遊戲工作階段的所有玩家工作階段。遊戲工作階段 ID 格式如下：`arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`。`<ID string>` 的值可能是自訂 ID 字串 (若在建立遊戲工作階段時有指定 ID)，或是產生的字串。

類型：字串

必要：否

限制

回傳結果的數量上限。使用此參數搭配 NextToken 取得結果做為一組循序頁面。若指定玩家工作階段 ID，此參數將遭到忽略。

類型：整數

必要：否

NextToken

字符顯示下一個結果循序頁面的開始處。使用前一個呼叫此動作傳回的字符。指定結果集的開始處時，請勿指定值。若指定玩家工作階段 ID，此參數將遭到忽略。

類型：字串

必要：否

PlayerId

玩家的唯一識別項。玩家 ID 是由開發人員定義。請參閱 [產生玩家 IDs](#)。

類型：字串

必要：否

PlayerSessionId

玩家工作階段的唯一識別項。

類型：字串

必要：否

PlayerSessionStatusFilter

用於篩選結果的玩家工作階段狀態。可能的玩家工作階段狀態包括下列項目：

- RESERVED – 玩家工作階段要求已收到，但玩家尚未連線至伺服器程序及/或通過驗證。
- ACTIVE – 玩家已由伺服器程序驗證，目前已連線。
- COMPLETED – 玩家連線已中斷。
- TIMEDOUT – 玩家工作階段要求已收到，但玩家並未在逾時限制 (60 秒) 內連線及/或通過驗證。

類型：字串

必要：否

LogParameters

此資料類型用於識別遊戲工作階段期間產生的哪些檔案要由 Amazon GameLift Servers 上傳，並在遊戲工作階段結束時存放。系統會在進行 [ProcessReady\(\)](#) 呼叫時，將此資訊傳送至 Amazon GameLift Servers 服務。

目錄

logPaths

您要 Amazon GameLift Servers 存放遊戲伺服器日誌檔案的目錄路徑，以供日後存取使用。這些檔案是在每個遊戲工作階段期間產生。檔案路徑及名稱於您的遊戲伺服器定義，並儲存於遊戲組建根目錄。日誌路徑必須是絕對的。例如若您的遊戲建構將遊戲工作階段記錄儲存於 MyGame\sessionlogs\ 這樣的目錄，則記錄路徑就可能是 c:\game\MyGame\sessionLogs (在 Windows 執行個體) 或 /local/game/MyGame/sessionLogs (在 Linux 執行個體)。

類型：std::vector<std::string>

必要：否

ProcessParameters

此資料類型包含一組參數，系統會在進行 [ProcessReady\(\)](#) 呼叫時將該組參數傳送至 Amazon GameLift Servers 服務。

目錄

port

伺服器程序接聽新玩家連線所在的連接埠編號。值必須屬於為部署此遊戲伺服器組建之機群所設定的連接埠範圍。此連接埠號碼包含在遊戲工作階段和遊戲工作階段物件中，遊戲工作階段會使用該物件來連接到伺服器程序。

類型：整數

必要：是

logParameters

含對遊戲工作階段日誌檔之目錄路徑清單的物件。

類型：Aws::GameLift::Server::[LogParameters](#)

必要：否

onStartGameSession

Amazon GameLift Servers 服務叫用以啟用新遊戲工作階段的回呼函數名稱。會Amazon GameLift Servers呼叫此函數以回應用戶端請求 [CreateGameSession](#)。該回呼函數會傳遞 [GameSession](#) 物件 (即 Amazon GameLift Servers 服務 API 參考中所定義的物件)。

類型：const std::function<void(Aws::GameLift::Model::GameSession)>
onStartGameSession

必要：是

onProcessTerminate

Amazon GameLift Servers 服務叫用以強制伺服器程序關閉的回呼函數名稱。呼叫此函數後，Amazon GameLift Servers 會等待五分鐘讓系統關閉伺服器程序，然後透過 [ProcessEnding\(\)](#) 呼叫進行回應。如果沒有收到回應，就會關閉伺服器程序。

類型：std::function<void()> onProcessTerminate

必要：否

onHealthCheck

Amazon GameLift Servers 服務叫用以從伺服器程序請求運作狀態報告的回呼函數名稱。每 60 秒 Amazon GameLift Servers 呼叫此函數。呼叫此函數後，Amazon GameLift Servers 會等待 60 秒確認是否有回應，若未收到任何回應，系統即會將該伺服器程序記錄為運作狀態不佳。

類型：std::function<bool()> onHealthCheck

必要：否

onUpdateGameSession

Amazon GameLift Servers 服務叫用以將更新的遊戲工作階段物件傳遞至伺服器程序的回呼函數名稱。會在處理[相符回填](#)請求時Amazon GameLift Servers呼叫此函數，以提供更新的配對建構器資料。然後傳送 [GameSession](#) 物件、狀態更新 (updateReason) 及配對回填票證 ID。

類

型：std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)>
onUpdateGameSession

必要：否

StartMatchBackfillRequest

此項資料類型用於傳送配對回填要求。系統會在進行 [StartMatchBackfill\(\)](#) 呼叫時，將此資訊傳送至 Amazon GameLift Servers 服務。

目錄

GameSessionArn

獨一無二的遊戲工作階段識別項。API 動作 [GetGameSessionId\(\)](#) 以 ARN 格式傳回識別項。

類型：字串

必要：是

MatchmakingConfigurationArn

以 ARN 為格式的唯一識別項，讓配對建構器使用此項要求。尋找用於建立原始遊戲工作階段的配對建構器時，請查看配對建構器資料屬性之中的遊戲工作階段物件。進一步了解 [Word 中的配對建構器資料與配對建構器資料](#)。

類型：字串

必要：是

Players

表示目前遊戲工作階段之中所有玩家的一組資料。配對建構器使用此項資訊搜尋適合配對現有玩家的新玩家。如需播放器物件格式的說明，請參閱 Amazon GameLift Servers API 參考指南。尋找玩家屬性、ID 及團隊指派時，請查看配對建構器資料屬性之中的遊戲工作階段物件。若配對建構器使用延遲，您可收集現有區域更新後的延遲，並將其納入各個玩家資料之中。

類型：std : vector[<player>](#)

必要：是

TicketId

配對或配對回填要求票證的唯一識別項。如果此處未提供任何值，Amazon GameLift Servers 將以 UUID 的形式產生一個值。您可使用此識別項依據需求追蹤配對回填票證狀態或取消要求。

類型：字串

必要：否

StopMatchBackfillRequest

此項資料類型用於取消配對回填要求。系統會在進行 [StopMatchBackfill\(\)](#) 呼叫時，將此資訊傳送至 Amazon GameLift Servers 服務。

目錄

GameSessionArn

與遭取消要求有關的唯一遊戲工作階段識別碼。

類型：字串

必要：是

MatchmakingConfigurationArn

做為此要求傳送目標的配對建構器唯一識別項。

類型：字串

必要：是

TicketId

遭取消回填要求票證的唯一識別碼。

類型：字串

必要：是

[適用於 Amazon GameLift Servers 4.x 的 C++ 伺服器 SDK -- 資料類型](#)

主題

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)

- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [Destroy\(\)](#)

AcceptPlayerSession()

通知Amazon GameLift Servers服務，具有指定玩家工作階段 ID 的玩家已連線至伺服器程序，且需要驗證。Amazon GameLift Servers 會驗證玩家工作階段 ID 是否有效，也就是玩家 ID 在遊戲工作階段中已預留玩家位置。經過驗證後，Amazon GameLift Servers 會將玩家位置的狀態從 RESERVED (已預留) 變更為 ACTIVE (使用中)。

語法

```
GenericOutcome AcceptPlayerSession(const std::string& playerSessionId);
```

參數

playerSessionId

由 Amazon GameLift Servers服務發出的唯一 ID，以回應對 AWS SDK Amazon GameLift Servers API 動作 [CreatePlayerSession](#) 的呼叫。遊戲用戶端在連線至伺服器程序時參考此 ID。

類型：std::string

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例展示了用來處理連線請求的函數，其處理過程包括驗證和拒絕無效的玩家工作階段 ID。

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
playerSessionId){
    Aws::GameLift::GenericOutcome connectOutcome =
        Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
    if(connectOutcome.IsSuccess())
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(connectOutcome.GetError().GetMessage());
    }
}
```

ActivateGameSession()

若伺服器程序已啟動遊戲工作階段，並且現在已準備好接受玩家連線，則此動作會通知 Amazon GameLift Servers 服務。此動作應當做 onStartGameSession() 回呼函數的一部分，在所有遊戲工作階段初始化完成後進行。

語法

```
GenericOutcome ActivateGameSession();
```

參數

此動作沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例顯示的是做為 onStartGameSession() 回呼函數一部分的 ActivateGameSession() 受到呼叫。

```
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession();
}
```

DescribePlayerSessions()

擷取玩家工作階段資料，包括設定、工作階段中繼資料和玩家資料。使用此動作可取得單一玩家工作階段資訊、一個遊戲工作階段中所有玩家工作階段的資訊，或是與單一玩家 ID 關聯的所有玩家工作階段資訊。

語法

```
DescribePlayerSessionsOutcome DescribePlayerSessions (
    const Aws::GameLift::Server::Model::DescribePlayerSessionsRequest
    &describePlayerSessionsRequest);
```

參數

describePlayerSessionsRequest

[DescribePlayerSessionsRequest](#) 物件描述的是要擷取哪個玩家工作階段。

必要：是

傳回值

如果成功，會傳回 `DescribePlayerSessionsOutcome` 物件，內含一組與請求參數相符的玩家工作階段物件。玩家工作階段物件的結構與 AWS SDK Amazon GameLift Servers API [PlayerSession](#) 資料類型相同。

範例

此範例展示了讓所有玩家工作階段均主動連線至指定之遊戲工作階段的請求。透過省略 `NextToken` 並將 `Limit` 值設為 10，Amazon GameLift Servers 會傳回前 10 組符合請求的玩家工作階段記錄。

```
// Set request parameters
Aws::GameLift::Server::Model::DescribePlayerSessionsRequest request;
```

```
request.SetPlayerSessionStatusFilter(Aws::GameLift::Server::Model::PlayerSessionStatusMapper::G  
request.SetLimit(10);  
request.SetGameSessionId("the game session ID");    // can use GetGameSessionId()  
  
// Call DescribePlayerSessions  
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =  
    Aws::GameLift::Server::DescribePlayerSessions(request);
```

GetGameSessionId()

若伺服器流程正在運作，擷取目前正在由伺服器程序託管的遊戲工作階段專屬識別符。此識別符會以 ARN 格式傳回：arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>。

對於尚未透過遊戲工作階段啟用的閒置程序，呼叫會傳回 Success= True 和 GameSessionId="" (空字串)。

語法

```
AwsStringOutcome GetGameSessionId();
```

參數

此動作沒有參數。

傳回值

如果成功，則會把遊戲工作階段 ID 當成 AwsStringOutcome 物件傳回。如果不成功，則會傳回錯誤訊息。

範例

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetInstanceCertificate()

擷取與機群及其執行個體相關聯的 pem 編碼 TLS 憑證的檔案位置。當您建立憑證組態設為 GENERATED 的新機群時，AWS Certificate Manager 會產生此憑證。使用此憑證可與遊戲用戶端建立安全連線，以及加密用戶端/伺服器通訊。

語法

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

參數

此動作沒有參數。

傳回值

如果成功，會傳回 `GetInstanceCertificateOutcome` 物件，其中包含存放在執行個體上的機群 TLS 憑證檔案和憑證鏈的位置。從憑證鏈擷取的根憑證檔案也會存放在執行個體上。如果不成功，則會傳回錯誤訊息。

如需憑證和憑證鏈資料的詳細資訊，請參閱 AWS Certificate Manager API 參考中的 [GetCertificate 回應元素](#)。

範例

```
Aws::GameLift::GetInstanceCertificateOutcome certificateOutcome =  
    Aws::GameLift::Server::GetInstanceCertificate();
```

GetSdkVersion()

傳回所用的 SDK 目前的版本編號。

語法

```
AwsStringOutcome GetSdkVersion();
```

參數

此動作沒有參數。

傳回值

如果成功，將目前開發套件版本以 `AwsStringOutcome` 物件傳回。傳回的字串僅包含版本編號（例如 "3.1.5"）。如果不成功，則會傳回錯誤訊息。

範例

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
```

```
Aws::GameLift::Server::GetSdkVersion();
```

GetTerminationTime()

若設有終止時間，即傳回伺服器程序排定關閉的時間。伺服器程序在收到 Amazon GameLift Servers 服務的回 `onProcessTerminate()` 呼後會採取此動作。Amazon GameLift Servers 可能會 `onProcessTerminate()` 呼叫，原因如下：(1) 伺服器程序回報運作狀態不佳或未回應 Amazon GameLift Servers、(2) 在縮減事件期間終止執行個體，或 (3) 執行個體因 [Spot 中斷](#) 而終止。

如果程序已收到回 `onProcessTerminate()` 呼，則傳回的值為預估的終止時間。如果程序尚未收到回 `onProcessTerminate()` 呼，則會傳回錯誤訊息。進一步了解 [關閉伺服器處理程序](#) 的相關資訊。

語法

```
AwsLongOutcome GetTerminationTime();
```

參數

此動作沒有參數。

傳回值

如果成功，會以 `AwsLongOutcome` 物件的形式傳回終止時間。該值是終止時間，以自 0001 00:00:00 起經過的刻度表示。例如，日期時間值 2020-09-13 12:26:40 -000Z 等於 637355968000000000 個刻度。如果沒有可用的終止時間，會傳回錯誤訊息。

範例

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

InitSDK()

初始化 Amazon GameLift Servers 開發套件。您應在啟動時隨即呼叫此方法，以避免系統先進行任何其他與 Amazon GameLift Servers 相關的初始化程序。

語法

```
InitSDKOutcome InitSDK();
```

參數

此動作沒有參數。

傳回值

如果成功，會傳回 `InitSdkOutcome` 物件，代表伺服器程序已準備好呼叫 [ProcessReady\(\)](#)。

範例

```
Aws::GameLift::Server::InitSDKOutcome initOutcome =  
    Aws::GameLift::Server::InitSDK();
```

ProcessEnding()

若伺服器程序正在關閉中，則此動作會通知 Amazon GameLift Servers 服務。此方法應於所有其他清除作業 (包括關閉所有作用中遊戲工作階段) 之後呼叫。此方法應以結束代碼 0 結束，非零的結束代碼會導致該程序未徹底結束的事件訊息出現。

一旦方法以代碼 0 結束，您就可以以成功的結束代碼終止程序。您也可以使用錯誤碼結束程序。如果您使用錯誤代碼結束，機群事件將指出程序異常終止 (`SERVER_PROCESS_TERMINATED_UNHEALTHY`)。

語法

```
GenericOutcome ProcessEnding();
```

參數

此動作沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();  
if (outcome.Success)  
    exit(0); // exit with success  
// otherwise, exit with error code  
exit(errorCode);
```


ProcessReady()

若伺服器程序已準備好託管遊戲工作階段，則此動作會通知 Amazon GameLift Servers 服務。在成功叫用 [InitSDK\(\)](#) 並完成伺服器程序可託管遊戲工作階段之前所需的設定任務之後，請呼叫此方法。每個程序只能呼叫此方法一次。

此為同步呼叫。若要進行非同步呼叫，請使用 [ProcessReadyAsync\(\)](#)。如需詳細資訊，請參閱[初始化伺服器程序](#)。

語法

```
GenericOutcome ProcessReady(  
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

參數

processParameters

[ProcessParameters](#) 物件會傳達以下有關伺服器程序的資訊：

- 在遊戲伺服器程式碼中實作的回呼方法名稱，可供 Amazon GameLift Servers 服務呼叫以與伺服器程序通訊。
- 伺服器程序正在接聽的埠號。
- 任何要 Amazon GameLift Servers 擷取並存放的遊戲工作階段特定檔案路徑。

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會說明 [ProcessReady\(\)](#) 呼叫和回呼函數的實作。

```
// Set parameters and call ProcessReady  
std::string serverLog("serverOut.log");           // Example of a log file written by the  
game server  
std::vector<std::string> logPaths;  
logPaths.push_back(serverLog);  
  
int listenPort = 9339;
```

```

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
        listenPort,
        Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcome outcome =
    Aws::GameLift::Server::ProcessReady(processReadyParameter);

// Implement callback functions
void Server::onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome =
        Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void Server::onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool Server::onHealthCheck()
{
    bool health;
    // complete health evaluation within 60 seconds and set health
    return health;
}

```

ProcessReadyAsync()

若伺服器程序已準備好託管遊戲工作階段，則此動作會通知 Amazon GameLift Servers 服務。一旦伺服器程序準備好代管遊戲工作階段時，即應呼叫此方法。該參數會指定 Amazon GameLift Servers 在特定情況下要呼叫的回呼函數名稱。遊戲伺服器代碼必須實作上述函數。

此為非同步呼叫。若要進行同步呼叫，請使用 [ProcessReady\(\)](#)。如需詳細資訊，請參閱[初始化伺服器程序](#)。

語法

```
GenericOutcomeCallable ProcessReadyAsync(  
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

參數

processParameters

[ProcessParameters](#) 物件會傳達以下有關伺服器程序的資訊：

- 在遊戲伺服器程式碼中實作的回呼方法名稱，可供 Amazon GameLift Servers 服務呼叫以與伺服器程序通訊。
- 伺服器程序正在接聽的埠號。
- 任何要 Amazon GameLift Servers 擷取並存放的遊戲工作階段特定檔案路徑。

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
// Set parameters and call ProcessReady  
std::string serverLog("serverOut.log");           // This is an example of a log file  
    written by the game server  
std::vector<std::string> logPaths;  
logPaths.push_back(serverLog);  
  
int listenPort = 9339;  
  
Aws::GameLift::Server::ProcessParameters processReadyParameter =  
    Aws::GameLift::Server::ProcessParameters(  
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),  
        std::bind(&Server::onProcessTerminate, this),  
        std::bind(&Server::OnHealthCheck, this),  
        std::bind(&Server::OnUpdateGameSession, this),  
        listenPort,  
        Aws::GameLift::Server::LogParameters(logPaths));  
  
Aws::GameLift::GenericOutcomeCallable outcome =
```

```

    Aws::GameLift::Server::ProcessReadyAsync(processReadyParameter);

// Implement callback functions
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool onHealthCheck()
{
    // perform health evaluation and complete within 60 seconds
    return health;
}

```

RemovePlayerSession()

若具有指定玩家工作階段 ID 的玩家已與伺服器程序中斷連線，則此動作會通知 Amazon GameLift Servers 服務。Amazon GameLift Servers 會對此做出回應，將玩家位置變更為可用，使該位置可指派給新玩家。

語法

```

GenericOutcome RemovePlayerSession(
    const std::string& playerSessionId);

```

參數

playerSessionId

由 Amazon GameLift Servers 服務發出的唯一 ID，以回應對 AWS SDK Amazon GameLift Servers API 動作 [CreatePlayerSession](#) 的呼叫。遊戲用戶端在連線至伺服器程序時參考此 ID。

類型：std::string

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

此動作會傳送請求，以便替 FlexMatch 所建立的遊戲工作階段開放空位找到新玩家。另請參閱 AWS SDK 動作 [StartMatchBackfill\(\)](#)。使用此動作，目前代管遊戲工作階段的遊戲伺服器程序即可初始化配對回填請求。進一步了解[FlexMatch 回填功能](#)。

此為非同步動作。如果新玩家成功配對，Amazon GameLift Servers 此服務會透過叫用回呼函數 來提供更新的配對建構器資料 `OnUpdateGameSession()`。

一個伺服器程序一次僅能有一個使用中的配對回填請求。若要發送新請求，請先呼叫 [StopMatchBackfill\(\)](#) 取消原始請求。

語法

```
StartMatchBackfillOutcome StartMatchBackfill (  
    const Aws::GameLift::Server::Model::StartMatchBackfillRequest  
    &startBackfillRequest);
```

參數

StartMatchBackfillRequest

[StartMatchBackfillRequest](#) 物件會傳達以下資訊：

- 指派給回填請求的票證 ID。此為選填的資訊，若未提供任何 ID，則 Amazon GameLift Servers 會自動產生一個。
- 傳送請求對象的配對建構器。必須填入完整的組態 ARN。此值可從遊戲工作階段的配對建構器資料中取得。
- 經回填之遊戲工作階段的 ID。
- 遊戲工作階段目前玩家可用的配對資料。

必要：是

傳回值

傳回 `StartMatchBackfillOutcome` 物件，附帶配對回填票證或含有錯誤訊息的故障狀況。您可以使用 AWS SDK 動作 [DescribeMatchmaking\(\)](#) 追蹤票證狀態。

範例

```
// Build a backfill request
std::vector<Player> players;
Aws::GameLift::Server::Model::StartMatchBackfillRequest startBackfillRequest;
startBackfillRequest.SetTicketId("a ticket ID");
    //optional, autogenerated if not provided
startBackfillRequest.SetMatchmakingConfigurationArn("the matchmaker configuration
    ARN"); //from the game session matchmaker data
startBackfillRequest.SetGameSessionArn("the game session ARN");
    // can use GetGameSessionId()
startBackfillRequest.SetPlayers(players);
    //from the game session matchmaker data

// Send backfill request
Aws::GameLift::StartMatchBackfillOutcome backfillOutcome =
    Aws::GameLift::Server::StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void Server::OnUpdateGameSession(Aws::GameLift::Server::Model::GameSession gameSession,
    Aws::GameLift::Server::Model::UpdateReason updateReason, std::string backfillTicketId)
{
    // handle status messages
    // perform game-specific tasks to prep for newly matched players
}
```

StopMatchBackfill()

取消以 [StartMatchBackfill\(\)](#) 建立的使用中配對回填請求。另請參閱 AWS SDK 動作 [StopMatchmaking\(\)](#)。進一步了解[FlexMatch回填功能](#)。

語法

```
GenericOutcome StopMatchBackfill (
    const Aws::GameLift::Server::Model::StopMatchBackfillRequest &stopBackfillRequest);
```

參數

StopMatchBackfillRequest

識別配對票證的 [StopMatchBackfillRequest](#) 物件，用以取消：

- 已取消指派給此回填請求的票證 ID
- 回填請求的傳送目標配對建構器
- 與回填請求相關的遊戲工作階段

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
// Set backfill stop request parameters

Aws::GameLift::Server::Model::StopMatchBackfillRequest stopBackfillRequest;
stopBackfillRequest.SetTicketId("the ticket ID");
stopBackfillRequest.SetGameSessionArn("the game session ARN");
// can use GetGameSessionId()
stopBackfillRequest.SetMatchmakingConfigurationArn("the matchmaker configuration ARN");
// from the game session matchmaker data

Aws::GameLift::GenericOutcome stopBackfillOutcome =
    Aws::GameLift::Server::StopMatchBackfillRequest(stopBackfillRequest);
```

TerminateGameSession()

此方法已棄用 4.0.1 版。相反地，伺服器程序應該[ProcessEnding\(\)](#)在遊戲工作階段結束後呼叫。

通知Amazon GameLift Servers服務伺服器程序已結束目前的遊戲工作階段。當伺服器程序保持作用中狀態並準備好託管新的遊戲工作階段時，就會呼叫此動作。只有在遊戲工作階段終止程序完成後，才應該呼叫它，因為它會向發出訊號Amazon GameLift Servers，表示伺服器程序可立即用於託管新的遊戲工作階段。

如果伺服器程序會在遊戲工作階段停止後關閉，則不會呼叫此動作。反之，呼叫 [ProcessEnding\(\)](#) 來表示遊戲工作階段和伺服器程序都已結束。

語法

```
GenericOutcome TerminateGameSession();
```

參數

此動作沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

UpdatePlayerSessionCreationPolicy()

更新目前遊戲工作階段的能力，以接受新的玩家工作階段。遊戲工作階段可設定為接受或拒絕所有新的玩家工作階段。另請參閱 SDK AWS 動作 [UpdateGameSession\(\)](#)。

語法

```
GenericOutcome UpdatePlayerSessionCreationPolicy(  
    Aws::GameLift::Model::PlayerSessionCreationPolicy newPlayerSessionPolicy);
```

參數

newPlayerSessionPolicy

字串值代表遊戲工作階段是否可接受新玩家。

類型：Aws::GameLift::Model::PlayerSessionCreationPolicy 列舉。有效值包含：

- ACCEPT_ALL – 接受所有新玩家工作階段。
- DENY_ALL – 拒絕所有新玩家工作階段。

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例設定目前遊戲工作階段的加入政策為可接受所有玩家。


```
Aws::GameLift::GenericOutcome outcome =  
    Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

Destroy()

清除遊戲伺服器初始化期間 initSDK() 配置的記憶體。在您結束遊戲伺服器程序後使用此方法，以避免浪費伺服器記憶體。

語法

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

參數

沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會在遊戲伺服器程序結束後清除 initSDK 配置的記憶體。

```
if (Aws::GameLift::Server::ProcessEnding().IsSuccess()) {  
    Aws::GameLift::Server::Destroy();  
    exit(0);  
}
```

適用於 Amazon GameLift Servers 4.x 的 C# 伺服器 SDK -- 動作

使用伺服器 SDK 參考整合多玩家遊戲，以便與進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

Note

此參考適用於的舊版伺服器 SDK Amazon GameLift Servers。如需最新版本，請參閱 [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)。

適用於 Amazon GameLift Servers 4.x 的 C# 伺服器 SDK -- 資料類型

使用伺服器 SDK 參考整合多玩家遊戲，以便與進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

Note

此參考適用於舊版的伺服器 SDK Amazon GameLift Servers。如需最新版本，請參閱 [適用於 -- 資料類型的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)。

[適用於 Amazon GameLift Servers 4.x 的 C# 伺服器 SDK -- 動作](#)

主題

- [LogParameters](#)
- [DescribePlayerSessionsRequest](#)
- [ProcessParameters](#)
- [StopMatchBackfillRequest](#)

LogParameters

此資料類型用於識別遊戲工作階段期間產生的哪些檔案要由 Amazon GameLift Servers 上傳，並在遊戲工作階段結束時存放。系統會在進行 [ProcessReady\(\)](#) 呼叫時，將此資訊傳送至 Amazon GameLift Servers 服務。

目錄

logPaths

列出您要 Amazon GameLift Servers 儲存的遊戲伺服器目錄路徑記錄檔案，以供未來存取使用。這些檔案是在每次遊戲工作階段期間由伺服器程序產生；檔案路徑及名稱於遊戲伺服器定義，並儲存於遊戲建構根目錄。日誌路徑必須是絕對的。例如若您的遊戲建構將遊戲工作階段記錄儲存於 MyGame\sessionlogs\ 這樣的目錄，則記錄路徑就可能是 c:\game\MyGame\sessionLogs (在 Windows 執行個體) 或 /local/game/MyGame/sessionLogs (在 Linux 執行個體)。

類型：List<String>

必要：否

DescribePlayerSessionsRequest

此資料類型用於指定要擷取的玩家工作階段，可透過多種方式使用：(1) 提供 `PlayerSessionId` 要求特定玩家工作階段；(2) 提供 `GameSessionId` 要求指定遊戲工作階段之中的所有玩家工作階段；或 (3) 提供 `PlayerId` 要求指定玩家的所有玩家工作階段。對於大量的玩家工作階段，可使用分頁參數擷取結果做為循序頁面。

目錄

GameSessionId

獨一無二的遊戲工作階段識別項。請使用此參數要求特定遊戲工作階段的所有玩家工作階段。遊戲工作階段 ID 格式如下：`arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`。`<ID string>` 的值可能是自訂 ID 字串 (若在建立遊戲工作階段時有指定 ID)，或是產生的字串。

類型：字串

必要：否

限制

回傳結果的數量上限。使用此參數搭配 `NextToken` 取得結果做為一組循序頁面。若指定玩家工作階段 ID，此參數將遭到忽略。

類型：整數

必要：否

NextToken

字符顯示下一個結果循序頁面的開始處。使用前一個呼叫此動作傳回的字符。指定結果集的開始處時，請勿指定值。若指定玩家工作階段 ID，此參數將遭到忽略。

類型：字串

必要：否

PlayerId

玩家的唯一識別項。玩家 ID 是由開發人員定義。請參閱 [產生玩家 IDs](#)。

類型：字串

必要：否

PlayerSessionId

玩家工作階段的唯一識別項。

類型：字串

必要：否

PlayerSessionStatusFilter

用於篩選結果的玩家工作階段狀態。可能的玩家工作階段狀態包括下列項目：

- RESERVED – 玩家工作階段要求已收到，但玩家尚未連線至伺服器程序及/或通過驗證。
- ACTIVE – 玩家已由伺服器程序驗證，目前已連線。
- COMPLETED – 玩家連線已中斷。
- TIMEDOUT – 玩家工作階段要求已收到，但玩家並未在逾時限制 (60 秒) 內連線及/或通過驗證。

類型：字串

必要：否

ProcessParameters

此資料類型包含一組參數，系統會在進行 [ProcessReady\(\)](#) 呼叫時將該組參數傳送至 Amazon GameLift Servers 服務。

目錄

port

伺服器處理的連接埠號碼將為新玩家連線進行接聽。值必須屬於為部署此遊戲伺服器組建之機群所設定的連接埠範圍。此連接埠號碼包含在遊戲工作階段和遊戲工作階段物件中，遊戲工作階段會使用該物件來連接到伺服器程序。

類型：整數

必要：是

logParameters

含對遊戲工作階段日誌檔之目錄路徑清單的物件。

類型：Aws::GameLift::Server::[LogParameters](#)

必要：是

onStartGameSession

Amazon GameLift Servers 服務叫用以啟用新遊戲工作階段的回呼函數名稱。會Amazon GameLift Servers呼叫此函數以回應用戶端請求 [CreateGameSession](#)。該回呼函數會擷取 [GameSession](#) 物件，其為 Amazon GameLift Servers 服務 API 參考所定義的物件。

類型：void OnStartGameSessionDelegate(GameSession gameSession)

必要：是

onProcessTerminate

Amazon GameLift Servers 服務叫用以強制伺服器程序關閉的回呼函數名稱。呼叫此函數後，Amazon GameLift Servers 會等待五分鐘，讓系統能關閉伺服器程序，並透過 [ProcessEnding\(\)](#) 呼叫進行回應，再將伺服器程序關閉。

類型：void OnProcessTerminateDelegate()

必要：是

onHealthCheck

Amazon GameLift Servers 服務叫用以向伺服器程序請求運作狀態報告的回呼函數名稱。每 60 秒 Amazon GameLift Servers 呼叫此函數。呼叫此函數後，Amazon GameLift Servers 會等待 60 秒確認是否有回應，若未收到任何回應，系統即會將該伺服器程序記錄為運作狀態不佳。

類型：bool OnHealthCheckDelegate()

必要：是

onUpdateGameSession

Amazon GameLift Servers 服務叫用以將更新後的遊戲工作階段物件傳遞至伺服器程序的回呼函數名稱。會在處理[相符回填](#)請求時Amazon GameLift Servers呼叫此函數，以提供更新的配對建構器資料。然後傳送 [GameSession](#) 物件、狀態更新 (updateReason) 及配對回填票證 ID。

類型：void OnUpdateGameSessionDelegate (UpdateGameSession
updateGameSession)

必要：否

StartMatchBackfillRequest

此項資料類型用於傳送配對回填要求。系統會在進行 [StartMatchBackfill\(\)](#) 呼叫時，將此資訊傳送至 Amazon GameLift Servers 服務。

目錄

GameSessionArn

獨一無二的遊戲工作階段識別項。SDK 方法會以 ARN 格式 [GetGameSessionId\(\)](#) 傳回識別符。

類型：字串

必要：是

MatchmakingConfigurationArn

以 ARN 為格式的唯一識別項，讓配對建構器使用此項要求。尋找用於建立原始遊戲工作階段的配對建構器時，請查看配對建構器資料屬性之中的遊戲工作階段物件。進一步了解使用配對建構器資料中的 [配對建構器資料](#)。

類型：字串

必要：是

Players

表示目前遊戲工作階段之中所有玩家的一組資料。配對建構器使用此項資訊搜尋適合配對現有玩家的新玩家。如需播放器物件格式的說明，請參閱 Amazon GameLift Servers API 參考指南。尋找玩家屬性、ID 及團隊指派時，請查看配對建構器資料屬性之中的遊戲工作階段物件。若配對建構器使用延遲，您可收集現有區域更新後的延遲，並將其納入各個玩家資料之中。

類型：[玩家](#)[]

必要：是

TicketId

配對或配對回填要求票證的唯一識別項。如果此處未提供任何值，Amazon GameLift Servers 將以 UUID 的形式產生一個值。您可使用此識別項依據需求追蹤配對回填票證狀態或取消要求。

類型：字串

必要：否

StopMatchBackfillRequest

此項資料類型用於取消配對回填要求。系統會在進行 [StopMatchBackfill\(\)](#) 呼叫時，將此資訊傳送至 Amazon GameLift Servers 服務。

目錄

GameSessionArn

與遭取消要求有關的唯一遊戲工作階段識別碼。

類型：字串

必要：是

MatchmakingConfigurationArn

做為此要求傳送目標的配對建構器唯一識別項。

類型：字串

必要：是

TicketId

遭取消回填要求票證的唯一識別碼。

類型：字串

必要：是

[適用於 Amazon GameLift Servers 4.x 的 C# 伺服器 SDK -- 資料類型](#)

主題

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)

- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

通知Amazon GameLift Servers服務，具有指定玩家工作階段 ID 的玩家已連線到伺服器程序，且需要驗證。Amazon GameLift Servers 會驗證玩家工作階段 ID 是否有效，也就是玩家 ID 在遊戲工作階段中已預留玩家位置。經過驗證後，Amazon GameLift Servers 會將玩家位置的狀態從 RESERVED (已預留) 變更為 ACTIVE (使用中)。

語法

```
GenericOutcome AcceptPlayerSession(String playerId)
```

參數

playerSessionId

建立新玩家工作階段Amazon GameLift Servers時由發出的唯一 ID。物件中會指定玩家工作階段 ID。PlayerSession，以回應用戶端對 GameLift API 動作 [StartGameSessionPlacement](#)、[CreateGameSession](#)、[DescribeGameSessionPlacement](#) 或 [DescribePlayerSessions](#) 的呼叫。

類型：字串

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例展示了用來處理連線請求的函數，其處理過程包括驗證和拒絕無效的玩家工作階段 ID。

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerId){
    var acceptPlayerSessionOutcome =
        GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
    if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerId);
        connection.Accept();
    }
    else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);
    }
}
```

ActivateGameSession()

通知 Amazon GameLift Servers 服務，伺服器程序已啟動遊戲工作階段，目前已準備好接受玩家連線。此動作應當做 onStartGameSession() 回呼函數的一部分，在所有遊戲工作階段初始化完成後進行。

語法

```
GenericOutcome ActivateGameSession()
```

參數

此動作沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例顯示的是做為 onStartGameSession() 委派函式一部分的 ActivateGameSession() 受到呼叫。

```
void OnStartGameSession(GameSession gameSession)
{
```

```
// game-specific tasks when starting a new game session, such as loading map

// When ready to receive players
var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

DescribePlayerSessions()

擷取玩家工作階段資料，包括設定、工作階段中繼資料和玩家資料。使用此動作可取得單一玩家工作階段資訊、一個遊戲工作階段中所有玩家工作階段的資訊，或是與單一玩家 ID 關聯的所有玩家工作階段資訊。

語法

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
describePlayerSessionsRequest)
```

參數

describePlayerSessionsRequest

[DescribePlayerSessionsRequest](#) 物件描述的是要擷取哪個玩家工作階段。

必要：是

傳回值

如果成功，會傳回 `DescribePlayerSessionsOutcome` 物件，內含一組與請求參數相符的玩家工作階段物件。玩家工作階段物件的結構與 AWS SDK Amazon GameLift Servers API [PlayerSession](#) 資料類型相同。

範例

此範例展示了讓所有玩家工作階段均主動連線至指定之遊戲工作階段的請求。省略 `NextToken`，並將 `Limit` 的值設定為 10 時，Amazon GameLift Servers 即會傳回前 10 組符合請求的玩家工作階段記錄。

```
// Set request parameters
var describePlayerSessionsRequest = new
    Aws.GameLift.Server.Model.DescribePlayerSessionsRequest()
{
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    //gets the ID for
    the current game session
}
```

```
    Limit = 10,
    PlayerSessionStatusFilter =
    PlayerSessionStatusMapper.GetNameForPlayerSessionStatus(PlayerSessionStatus.ACTIVE)
};
// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::Model::DescribePlayerSessions(describePlayerSessionRequest);
```

GetGameSessionId()

若伺服器流程正在運作，擷取目前正在由伺服器程序託管的遊戲工作階段 ID。

對於尚未透過遊戲工作階段啟用的閒置程序，呼叫會傳回 Success=True 和 GameSessionId="" (空字串)。

語法

```
AwsStringOutcome GetGameSessionId()
```

參數

此動作沒有參數。

傳回值

如果成功，則會把遊戲工作階段 ID 當成 AwsStringOutcome 物件傳回。如果不成功，則會傳回錯誤訊息。

範例

```
var getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetInstanceCertificate()

擷取與機群及其執行個體相關聯的 pem 編碼 TLS 憑證的檔案位置。當您建立憑證組態設為 GENERATED 的新機群時，AWS Certificate Manager 會產生此憑證。使用此憑證可與遊戲用戶端建立安全連線，以及加密用戶端/伺服器通訊。

語法

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

參數

此動作沒有參數。

傳回值

如果成功，會傳回 `GetInstanceCertificateOutcome` 物件，其中包含存放在執行個體上的機群 TLS 憑證檔案和憑證鏈的位置。從憑證鏈擷取的根憑證檔案也會存放在執行個體上。如果不成功，則會傳回錯誤訊息。

如需憑證和憑證鏈資料的詳細資訊，請參閱 AWS Certificate Manager API 參考中的 [GetCertificate 回應元素](#)。

範例

```
var getInstanceCertificateOutcome = GameLiftServerAPI.GetInstanceCertificate();
```

GetSdkVersion()

傳回內建至伺服器程序的目前開發套件版本編號。

語法

```
AwsStringOutcome GetSdkVersion()
```

參數

此動作沒有參數。

傳回值

如果成功，將目前開發套件版本以 `AwsStringOutcome` 物件傳回。傳回的字串僅包含版本編號（例如 "3.1.5"）。如果不成功，則會傳回錯誤訊息。

範例

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

GetTerminationTime()

若設有終止時間，即傳回伺服器程序排定關閉的時間。伺服器程序在收到 Amazon GameLift Servers 服務的回 `onProcessTerminate()` 呼後會採取此動作。Amazon GameLift Servers 可能

會 `onProcessTerminate()` 因為下列原因呼叫：(1) 運作狀態不佳（伺服器程序已回報連接埠運作狀態或尚未回應 Amazon GameLift Servers、(2) 在縮減規模事件期間終止執行個體時，或 (3) 執行個體因 [spot-instance 中斷](#) 而終止時。

如果程序已收到回 `onProcessTerminate()` 呼，則傳回的值為預估的終止時間。如果程序尚未收到回 `onProcessTerminate()` 呼，則會傳回錯誤訊息。進一步了解 [關閉伺服器處理程序](#) 的相關資訊。

語法

```
AwsDateTimeOutcome GetTerminationTime()
```

參數

此動作沒有參數。

傳回值

如果成功，會將終止時間傳回為 `AwsDateTimeOutcome` 物件。該值是終止時間，以自 0001 00:00:00 起已經過的刻度表示。例如，日期時間值 2020-09-13 12:26:40 -000Z 等於 637355968000000000 個刻度。如果沒有可用的終止時間，會傳回錯誤訊息。

範例

```
var getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

InitSDK()

初始化 Amazon GameLift Servers 開發套件。您應在啟動時隨即呼叫此方法，以避免系統先進行任何其他與 Amazon GameLift Servers 相關的初始化程序。

語法

```
InitSDKOutcome InitSDK()
```

參數

此動作沒有參數。

傳回值

如果成功，會傳回 `InitSdkOutcome` 物件，代表伺服器程序已準備好呼叫 [ProcessReady\(\)](#)。

範例

```
var initSDKOutcome = GameLiftServerAPI.InitSDK();
```

ProcessEnding()

若伺服器程序正在關閉中，則此動作會通知 Amazon GameLift Servers 服務。此方法應於所有其他清除作業 (包括關閉所有作用中遊戲工作階段) 之後呼叫。此方法應以結束代碼 0 結束，非零的結束代碼會導致該程序未徹底結束的事件訊息出現。

一旦方法以代碼 0 結束，您就可以以成功的結束代碼終止程序。您也可以使用錯誤碼結束程序。如果您使用錯誤代碼結束，機群事件會指出程序異常終止 (SERVER_PROCESS_TERMINATED_UNHEALTHY)。

語法

```
GenericOutcome ProcessEnding()
```

參數

此動作沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
    Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

ProcessReady()

若伺服器程序已準備好託管遊戲工作階段，則此動作會通知 Amazon GameLift Servers 服務。在成功叫用 [InitSDK\(\)](#) 並完成伺服器程序可以託管遊戲工作階段之前所需的設定任務之後，呼叫此方法。每個程序只能呼叫此方法一次。

語法

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

參數

processParameters

[ProcessParameters](#) 物件會傳達以下有關伺服器程序的資訊：

- 在遊戲伺服器程式碼中實作的回呼方法名稱，可供 Amazon GameLift Servers 服務呼叫以與伺服器程序通訊。
- 伺服器程序正在接聽的埠號。
- 任何要 Amazon GameLift Servers 擷取並存放的遊戲工作階段特定檔案路徑。

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例會說明 [ProcessReady\(\)](#) 呼叫和委派函數的實作。

```
// Set parameters and call ProcessReady
var processParams = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    new LogParameters(new List<string>()           // Examples of log and error files
        written by the game server
        {
            "C:\\game\\logs",
            "C:\\game\\error"
        })
);

var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

```
// Implement callback functions
void OnGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}

void OnProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    var ProcessEndingOutcome = GameLiftServerAPI.ProcessEnding();
}

bool OnHealthCheck()
{
    bool isHealthy;
    // complete health evaluation within 60 seconds and set health
    return isHealthy;
}
```

RemovePlayerSession()

若具有指定玩家工作階段 ID 的玩家已與伺服器程序中斷連線，則此動作會通知 Amazon GameLift Servers 服務。Amazon GameLift Servers 會對此做出回應，將玩家位置變更為可用，使該位置可指派給新玩家。

語法

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

參數

playerSessionId

建立新玩家工作階段 Amazon GameLift Servers 時由發出的唯一 ID。物件中會指定玩家工作階段 ID `PlayerSession`，以回應用戶端對 GameLift API 動作 [StartGameSessionPlacement](#)、[CreateGameSession](#)、[DescribeGameSessionPlacement](#) 或 [DescribePlayerSessions](#) 的呼叫。

類型：字串

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

此動作會傳送請求，以便替 FlexMatch 所建立的遊戲工作階段開放空位找到新玩家。另請參閱 AWS SDK 動作 [StartMatchBackfill\(\)](#)。使用此動作，目前代管遊戲工作階段的遊戲伺服器程序即可初始化配對回填請求。進一步了解[FlexMatch回填功能](#)。

此為非同步動作。如果新玩家配對成功，Amazon GameLift Servers 服務即會使用回呼函數 `OnUpdateGameSession()` 提供更新的配對建構器資料。

一個伺服器程序一次僅能有一個使用中的配對回填請求。若要發送新請求，請先呼叫 [StopMatchBackfill\(\)](#) 取消原始請求。

語法

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest  
    startBackfillRequest);
```

參數

StartMatchBackfillRequest

[StartMatchBackfillRequest](#) 物件會傳達以下資訊：

- 指派給回填請求的票證 ID。此為選填的資訊，若未提供任何 ID，則 Amazon GameLift Servers 會自動產生一個。
- 傳送請求對象的配對建構器。必須填入完整的組態 ARN。此值可從遊戲工作階段的配對建構器資料中取得。
- 經回填之遊戲工作階段的 ID。
- 遊戲工作階段目前玩家可用的配對資料。

必要：是

傳回值

傳回 StartMatchBackfillOutcome 物件，附帶配對回填票證 ID 或含有錯誤訊息的故障狀況。

範例

```
// Build a backfill request
var startBackfillRequest = new AWS.GameLift.Server.Model.StartMatchBackfillRequest()
{
    TicketId = "a ticket ID", //optional
    MatchmakingConfigurationArn = "the matchmaker configuration ARN",
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result, // gets ID for
current game session
    //get player data for all currently connected players
    MatchmakerData matchmakerData =
        MatchmakerData.FromJson(gameSession.MatchmakerData); // gets matchmaker
data for current players
    // get matchmakerData.Players
    // remove data for players who are no longer connected
    Players = ListOfPlayersRemainingInTheGame
};

// Send backfill request
var startBackfillOutcome = GameLiftServerAPI.StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void OnUpdateGameSession(GameSession myGameSession)
{
    // game-specific tasks to prepare for the newly matched players and update
    matchmaker data as needed
}
```

StopMatchBackfill()

取消以 [StartMatchBackfill\(\)](#) 建立的使用中配對回填請求。另請參閱 AWS SDK 動作 [StopMatchmaking\(\)](#)。進一步了解[FlexMatch回填功能](#)。

語法

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

參數

StopMatchBackfillRequest

識別配對票證的 [StopMatchBackfillRequest](#) 物件，用以取消：

- 已取消指派給此回填請求的票證 ID
- 回填請求的傳送目標配對建構器
- 與回填請求相關的遊戲工作階段

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

```
// Set backfill stop request parameters

var stopBackfillRequest = new AWS.GameLift.Server.Model.StopMatchBackfillRequest()
{
    TicketId = "a ticket ID", //optional, if not provided one is autogenerated
    MatchmakingConfigurationArn = "the matchmaker configuration ARN", //from the game
    session matchmaker data
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result //gets the ID for
    the current game session
};

var stopBackfillOutcome =
    GameLiftServerAPI.StopMatchBackfillRequest(stopBackfillRequest);
```

TerminateGameSession()

此方法已棄用 4.0.1 版。相反地，伺服器程序應該[ProcessEnding\(\)](#)在遊戲工作階段結束後呼叫。

通知Amazon GameLift Servers服務伺服器程序已結束目前的遊戲工作階段。當伺服器程序保持作用中狀態並準備好託管新的遊戲工作階段時，就會呼叫此動作。只有在遊戲工作階段終止程序完成後，才應該呼叫它，因為它會向發出訊號Amazon GameLift Servers，表示伺服器程序可立即用於託管新的遊戲工作階段。

如果伺服器程序會在遊戲工作階段停止後關閉，則不會呼叫此動作。反之，請呼叫 [ProcessEnding\(\)](#) 來表示遊戲工作階段和伺服器程序都即將結束。

語法

```
GenericOutcome TerminateGameSession()
```

參數

此動作沒有參數。

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例說明遊戲工作階段結束時的伺服器程序。

```
// game-specific tasks required to gracefully shut down a game session,  
// such as notifying players, preserving game state data, and other cleanup  
  
var terminateGameSessionOutcome = GameLiftServerAPI.TerminateGameSession();  
var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

UpdatePlayerSessionCreationPolicy()

更新目前遊戲工作階段的能力，以接受新的玩家工作階段。遊戲工作階段可設定為接受或拒絕所有新的玩家工作階段。(也請參閱 https://docs.aws.amazon.com/gamelift/latest/apireference/API_UpdateGameSession.html 服務 API 參考之中的 Amazon GameLift ServersUpdateGameSession() 動作)。

語法

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy  
playerSessionPolicy)
```

參數

newPlayerSessionPolicy

字串值代表遊戲工作階段是否可接受新玩家。

類型：[PlayerSessionCreationPolicy](#) 列舉。有效值包含：

- ACCEPT_ALL – 接受所有新玩家工作階段。
- DENY_ALL – 拒絕所有新玩家工作階段。

必要：是

傳回值

傳回一般結果，其中包含成功或失敗以及錯誤訊息。

範例

此範例設定目前遊戲工作階段的加入政策為可接受所有玩家。

```
var updatePlayerSessionCreationPolicyOutcomex =  
  
    GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

適用於 -- Amazon GameLift Servers 動作的伺服器 SDK (Unreal)

使用適用於 Unreal 的伺服器 SDK 整合多玩家遊戲，以便與進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

Note

此參考適用於舊版的伺服器 SDK Amazon GameLift Servers。如需最新版本，請參閱 [適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x](#)。

此 API 定義於 GameLiftServerSDK.h 和 GameLiftServerSDKModels.h。

若要設定 Unreal Engine 外掛程式，並查看程式碼範例 [Amazon GameLift Servers 整合至 Unreal Engine 專案](#)。

適用於 -- 資料類型的伺服器 SDK Amazon GameLift Servers (Unreal)

使用適用於 Unreal 的 Amazon GameLift Servers 伺服器 SDK 參考，整合您的多玩家遊戲以與進行託管 Amazon GameLift Servers。如需整合程序的指引，請參閱 [Amazon GameLift Servers 新增至您的遊戲伺服器](#)。

 Note

此參考適用於舊版的伺服器 SDK Amazon GameLift Servers。如需最新版本，請參閱[適用於 -- 資料類型的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x](#)。

此 API 定義於 `GameLiftServerSDK.h` 和 `GameLiftServerSDKModels.h`。

若要設定 Unreal Engine 外掛程式，並查看程式碼範例 [Amazon GameLift Servers 整合至 Unreal Engine 專案](#)。

[適用於 -- Amazon GameLift Servers 動作的伺服器 SDK \(Unreal\)](#)

主題

- [FDescribePlayerSessionsRequest](#)
- [FProcessParameters](#)
- [FStartMatchBackfillRequest](#)
- [FStopMatchBackfillRequest](#)

FDescribePlayerSessionsRequest

此資料類型用於指定要擷取的玩家工作階段，您可利用下列方式使用：

- 提供 `PlayerSessionId` 以要求特定玩家工作階段。
- 提供 `GameSessionId` 以要求特定遊戲工作階段之中的所有玩家工作階段。
- 提供 `PlayerId` 以要求特定玩家的所有玩家工作階段。

對於大量的玩家工作階段，可使用分頁參數於循序區塊擷取結果。

目錄

GameSessionId

獨一無二的遊戲工作階段識別項。請使用此參數要求特定遊戲工作階段的所有玩家工作階段。遊戲工作階段 ID 格式如下：`arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`。`<ID string>` 的值可能是自訂 ID 字串（若在建立遊戲工作階段時有指定 ID），或是產生的字串。

類型：字串

必要：否

限制

回傳結果的數量上限。使用此參數搭配 NextToken 取得結果做為一組循序頁面。若指定玩家工作階段 ID，此參數將遭到忽略。

類型：整數

必要：否

NextToken

字符顯示下一個結果循序頁面的開始處。使用前一個呼叫此動作傳回的字符。指定結果集的開始處時，請勿指定值。若指定玩家工作階段 ID，此參數將遭到忽略。

類型：字串

必要：否

PlayerId

玩家的唯一識別項。玩家 ID 是由開發人員定義。請參閱 [產生玩家 IDs](#)。

類型：字串

必要：否

PlayerSessionId

玩家工作階段的唯一識別項。

類型：字串

必要：否

PlayerSessionStatusFilter

用於篩選結果的玩家工作階段狀態。可能的玩家工作階段狀態包括下列項目：

- RESERVED – 玩家工作階段要求已收到，但玩家尚未連線至伺服器程序及/或通過驗證。
- ACTIVE – 玩家已由伺服器程序驗證，目前已連線。
- COMPLETED – 玩家連線已中斷。
- TIMEDOUT – 玩家工作階段要求已收到，但玩家並未在逾時限制 (60 秒) 內連線及/或通過驗證。

類型：字串

必要：否

FProcessParameters

此資料類型包含一組參數，系統會在進行 [ProcessReady\(\)](#) 呼叫時將該組參數傳送至 Amazon GameLift Servers 服務。

目錄

port

伺服器處理的連接埠號碼將為新玩家連線進行接聽。值必須屬於為部署此遊戲伺服器組建之機群所設定的連接埠範圍。此連接埠號碼包含在遊戲工作階段和遊戲工作階段物件中，遊戲工作階段會使用該物件來連接到伺服器程序。

類型：整數

必要：是

logParameters

含對遊戲工作階段日誌檔之目錄路徑清單的物件。

類型：TArray<FString>

必要：否

onStartGameSession

Amazon GameLift Servers 服務叫用以啟用新遊戲工作階段的回呼函數名稱。會Amazon GameLift Servers呼叫此函數以回應用戶端請求 [CreateGameSession](#)。該回呼函數會擷取 [GameSession](#) 物件，其為 Amazon GameLift Servers 服務 API 參考所定義的物件。

類型：FOnStartGameSession

必要：是

onProcessTerminate

Amazon GameLift Servers 服務叫用以強制伺服器程序關閉的回呼函數名稱。呼叫此函數後，Amazon GameLift Servers 會等待五分鐘，讓系統能關閉伺服器程序，並透過 [ProcessEnding\(\)](#) 呼叫進行回應，再將伺服器程序關閉。

類型：FSimpleDelegate

必要：否

onHealthCheck

Amazon GameLift Servers 服務叫用以向伺服器程序請求運作狀態報告的回呼函數名稱。每 60 秒 Amazon GameLift Servers 呼叫此函數。呼叫此函數後，Amazon GameLift Servers 會等待 60 秒確認是否有回應，若未收到任何回應，系統即會將該伺服器程序記錄為運作狀態不佳。

類型：FOnHealthCheck

必要：否

onUpdateGameSession

Amazon GameLift Servers 服務叫用以將更新的遊戲工作階段物件傳遞至伺服器程序的回呼函數名稱。會在處理[相符回填](#)請求時 Amazon GameLift Servers 呼叫此函數，以提供更新的配對建構器資料。然後傳送 [GameSession](#) 物件、狀態更新 (updateReason) 及配對回填票證 ID。

類型：FOnUpdateGameSession

必要：否

FStartMatchBackfillRequest

此項資料類型用於傳送配對回填要求。系統會在進行 [StartMatchBackfill\(\)](#) 呼叫時，將此資訊傳送至 Amazon GameLift Servers 服務。

目錄

GameSessionArn

獨一無二的遊戲工作階段識別項。API 動作 [GetGameSessionId\(\)](#) 以 ARN 格式傳回識別項。

類型：FString

必要：是

MatchmakingConfigurationArn

以 ARN 為格式的唯一識別項，讓配對建構器使用此項要求。尋找用於建立原始遊戲工作階段的配對建構器時，請查看配對建構器資料屬性之中的遊戲工作階段物件。進一步了解使用配對建構器資料中的[配對建構器資料](#)。

類型：FString

必要：是

Players

表示目前遊戲工作階段之中所有玩家的一組資料。配對建構器使用此項資訊搜尋適合配對現有玩家的新玩家。如需播放器物件格式的說明，請參閱 Amazon GameLift Servers API 參考指南。尋找玩家屬性、ID 及團隊指派時，請查看配對建構器資料屬性之中的遊戲工作階段物件。若配對建構器使用延遲，您可收集現有區域更新後的延遲，並將其納入各個玩家資料之中。

類型：TArray<[FPlayer](#)>

必要：是

TicketId

配對或配對回填要求票證的唯一識別項。如果此處未提供任何值，Amazon GameLift Servers 將以 UUID 的形式產生一個值。您可使用此識別項依據需求追蹤配對回填票證狀態或取消要求。

類型：FString

必要：否

FStopMatchBackfillRequest

此項資料類型用於取消配對回填要求。系統會在進行 [StopMatchBackfill\(\)](#) 呼叫時，將此資訊傳送至 Amazon GameLift Servers 服務。

目錄

GameSessionArn

與遭取消要求有關的唯一遊戲工作階段識別碼。

類型：FString

必要：是

MatchmakingConfigurationArn

做為此要求傳送目標的配對建構器唯一識別項。

類型：FString

必要：是

TicketId

遭取消回填要求票證的唯一識別碼。

類型：FString

必要：是

適用於 -- 資料類型的伺服器 SDK Amazon GameLift Servers (Unreal)

主題

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

通知Amazon GameLift Servers服務，具有指定玩家工作階段 ID 的玩家已連線至伺服器程序，且需要驗證。Amazon GameLift Servers 會驗證玩家工作階段 ID 是否有效，也就是玩家 ID 已在遊戲工作階

段中預留玩家位置。經過驗證後，Amazon GameLift Servers 會將玩家位置的狀態從 RESERVED (已預留) 變更為 ACTIVE (使用中)。

語法

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

參數

playerSessionId

由 Amazon GameLift Servers 服務發出的唯一 ID，以回應對 AWS SDK Amazon GameLift Servers API 動作 [CreatePlayerSession](#) 的呼叫。遊戲用戶端會在連線至伺服器程序時參考此 ID。

類型：FString

必要：是

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

ActivateGameSession()

通知 Amazon GameLift Servers 服務，伺服器程序已啟動遊戲工作階段，目前已準備好接受玩家連線。此動作應當做 onStartGameSession() 回呼函數的一部分，在所有遊戲工作階段初始化完成後進行。

語法

```
FGameLiftGenericOutcome ActivateGameSession()
```

參數

此動作沒有參數。

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

DescribePlayerSessions()

擷取玩家工作階段資料，包括設定、工作階段中繼資料和玩家資料。使用此動作可取得單一玩家工作階段資訊、一個遊戲工作階段中所有玩家工作階段的資訊，或是與單一玩家 ID 關聯的所有玩家工作階段資訊。

語法

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const  
    FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

參數

describePlayerSessionsRequest

[FDescribePlayerSessionsRequest](#) 物件描述的是要擷取哪個玩家工作階段。

必要：是

傳回值

如果成功，會傳回 [FDescribePlayerSessionsRequest](#) 物件，內含一組與請求參數相符的玩家工作階段物件。玩家工作階段物件的結構與 AWS SDK Amazon GameLift Servers API [PlayerSession](#) 資料類型相同。

GetGameSessionId()

若伺服器流程正在運作，擷取目前正在由伺服器程序託管的遊戲工作階段 ID。

語法

```
FGameLiftStringOutcome GetGameSessionId()
```

參數

此動作沒有參數。

傳回值

如果成功，則會把遊戲工作階段 ID 當成 FGameLiftStringOutcome 物件傳回。如果不成功，則會傳回錯誤訊息。

GetInstanceCertificate()

擷取與機群及其執行個體相關聯的 pem 編碼 TLS 憑證檔案位置。當您建立憑證組態設為 GENERATED 的新機群時，AWS Certificate Manager 會產生此憑證。使用此憑證可與遊戲用戶端建立安全連線，以及加密用戶端/伺服器通訊。

語法

```
FGameLiftGetInstanceCertificateOutcome GetInstanceCertificate()
```

參數

此動作沒有參數。

傳回值

如果成功，會傳回GetInstanceCertificateOutcome物件，其中包含機群的 TLS 憑證檔案和憑證鏈的位置，這些檔案存放在執行個體上。從憑證鏈擷取的根憑證檔案也會存放在執行個體上。如果不成功，則會傳回錯誤訊息。

如需憑證和憑證鏈資料的詳細資訊，請參閱 AWS Certificate Manager API 參考中的 [GetCertificate 回應元素](#)。

GetSdkVersion()

傳回內建至伺服器程序的目前開發套件版本編號。

語法

```
FGameLiftStringOutcome GetSdkVersion();
```

參數

此動作沒有參數。

傳回值

如果成功，將目前開發套件版本以 FGameLiftStringOutcome 物件傳回。傳回的字串僅包含版本編號（例如 "3.1.5"）。如果不成功，則會傳回錯誤訊息。

範例

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

初始化 Amazon GameLift Servers 開發套件。您應在啟動時隨即呼叫此方法，以避免系統先進行任何其他與 Amazon GameLift Servers 相關的初始化程序。

語法

```
FGameLiftGenericOutcome InitSDK()
```

參數

此動作沒有參數。

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

ProcessEnding()

若伺服器程序正在關閉中，則此動作會通知 Amazon GameLift Servers 服務。此方法應於所有其他清除作業 (包括關閉所有作用中遊戲工作階段) 之後呼叫。此方法應以結束代碼 0 結束，非零的結束代碼會導致該程序未徹底結束的事件訊息出現。

語法

```
FGameLiftGenericOutcome ProcessEnding()
```

參數

此動作沒有參數。

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

ProcessReady()

若伺服器程序已準備好託管遊戲工作階段，則此動作會通知 Amazon GameLift Servers 服務。在成功叫用 [InitSDK\(\)](#) 並完成伺服器程序可託管遊戲工作階段之前所需的設定任務之後，呼叫此方法。每個程序只能呼叫此方法一次。

語法

```
FGameLiftGenericOutcome ProcessReady(FProcessParameters &processParameters)
```

參數

FProcessParameters

[FProcessParameters](#) 物件會傳達以下有關伺服器程序的資訊：

- 在遊戲伺服器程式碼中實作的回呼方法名稱，可供 Amazon GameLift Servers 服務呼叫以與伺服器程序通訊。
- 伺服器程序正在接聽的埠號。
- 任何要 Amazon GameLift Servers 擷取並存放的遊戲工作階段特定檔案路徑。

必要：是

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

範例

請在[使用 Unreal Engine 外掛程式](#)檢視範本程式碼。

RemovePlayerSession()

若具有指定玩家工作階段 ID 的玩家已與伺服器程序中斷連線，則此動作會通知 Amazon GameLift Servers 服務。Amazon GameLift Servers 會對此做出回應，將玩家位置變更為可用，使該位置可指派給新玩家。

語法

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerId)
```


參數

playerSessionId

由 Amazon GameLift Servers 服務發出的唯一 ID，以回應對 AWS SDK Amazon GameLift Servers API 動作 [CreatePlayerSession](#) 的呼叫。遊戲用戶端會在連線至伺服器程序時參考此 ID。

類型：FString

必要：是

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

StartMatchBackfill()

此動作會傳送請求，以便替 FlexMatch 所建立的遊戲工作階段開放空位找到新玩家。另請參閱 AWS SDK 動作 [StartMatchBackfill\(\)](#)。使用此動作，目前代管遊戲工作階段的遊戲伺服器程序即可初始化配對回填請求。進一步了解[FlexMatch 回填功能](#)。

此為非同步動作。如果新玩家配對成功，Amazon GameLift Servers 服務即會使用回呼函數 `OnUpdateGameSession()` 提供更新的配對建構器資料。

一個伺服器程序一次僅能有一個使用中的配對回填請求。若要發送新請求，請先呼叫 [StopMatchBackfill\(\)](#) 取消原始請求。

語法

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest  
&startBackfillRequest);
```

參數

FStartMatchBackfillRequest

[FStartMatchBackfillRequest](#) 物件會傳達以下資訊：

- 指派給回填請求的票證 ID。此為選填的資訊，若未提供任何 ID，則 Amazon GameLift Servers 會自動產生一個。
- 傳送請求對象的配對建構器。必須填入完整的組態 ARN。此值可從遊戲工作階段的配對建構器資料中取得。

- 經回填之遊戲工作階段的 ID。
- 遊戲工作階段目前玩家可用的配對資料。

必要：是

傳回值

如果成功，會傳回符合的回填票證做為 `FGameLiftStringOutcome` 物件。如果不成功，則會傳回錯誤訊息。您可以使用 AWS SDK 動作 [DescribeMatchmaking\(\)](#) 追蹤票證狀態。

StopMatchBackfill()

取消以 [StartMatchBackfill\(\)](#) 建立的使用中配對回填請求。另請參閱 AWS SDK 動作 [StopMatchmaking\(\)](#)。進一步了解[FlexMatch回填功能](#)。

語法

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest  
&stopBackfillRequest);
```

參數

StopMatchBackfillRequest

識別配對票證的 [FStopMatchBackfillRequest](#) 物件，用以取消：

- 已取消指派給此回填請求的票證 ID
- 回填請求的傳送目標配對建構器
- 與回填請求相關的遊戲工作階段

必要：是

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

TerminateGameSession()

此方法已棄用 4.0.1 版。反之，伺服器程序應該[ProcessEnding\(\)](#)在遊戲工作階段結束後呼叫。

通知Amazon GameLift Servers服務伺服器程序已結束目前的遊戲工作階段。當伺服器程序保持作用中狀態並準備好託管新的遊戲工作階段時，就會呼叫此動作。只有在遊戲工作階段終止程序完成後，才應該呼叫它，因為它會向 發出訊號Amazon GameLift Servers，表示伺服器程序可立即用於託管新的遊戲工作階段。

如果在遊戲工作階段停止後伺服器程序將會關閉，則不會呼叫此動作。反之，請呼叫 [ProcessEnding\(\)](#) 以發出遊戲工作階段和伺服器程序都即將結束的訊號。

語法

```
FGameLiftGenericOutcome TerminateGameSession()
```

參數

此動作沒有參數。

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

UpdatePlayerSessionCreationPolicy()

更新目前遊戲工作階段的能力，以接受新的玩家工作階段。遊戲工作階段可設定為接受或拒絕所有新的玩家工作階段。(也請參閱 [UpdateGameSession\(\) 服務 API 參考](#) 之中的 Amazon GameLift Servers 動作)。

語法

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy policy)
```

參數

政策

值代表遊戲工作階段是否接受新玩家。

類型：EPlayerSessionCreationPolicy enum。有效值包含：

- ACCEPT_ALL – 接受所有新玩家工作階段。
- DENY_ALL – 拒絕所有新玩家工作階段。

必要：是

傳回值

傳回包含成功或失敗與錯誤訊息的一般結果。

Amazon GameLift ServersAmazon GameLift Servers參考Realtime

本節包含 Amazon GameLift Servers Amazon GameLift ServersRealtime SDK 的參考文件。它包含 Realtime用戶端 API 以及設定Amazon GameLift ServersRealtime指令碼的指引。

主題

- [Amazon GameLift ServersRealtime 用戶端 API \(C#\) 參考](#)
- [的指令碼參考 Amazon GameLift ServersRealtime](#)

Amazon GameLift ServersRealtime 用戶端 API (C#) 參考

使用Realtime用戶端 API 準備您的多玩家遊戲用戶端以搭配 使用Amazon GameLift ServersAmazon GameLift ServersRealtime。用戶端 API 包含一組同步 API 呼叫和非同步回呼，可讓遊戲用戶端連接到 Realtime 伺服器，並透過伺服器與其他遊戲用戶端交換訊息和資料。

此 API 已在下列程式庫中定義：

Client.cs

- [同步動作](#)
- [非同步回呼](#)
- [資料類型](#)

設定Realtime用戶端 API

1. 下載[Amazon GameLift ServersRealtime用戶端 SDK](#)。
2. 建置 C# 開發套件程式庫。找到解決方案檔案 GameLiftRealtimeClientSdkNet45.sln。請參閱 README.md 檔案，了解適用於最低需求和其他建置選項的 C# 伺服器開發套件。在 IDE 中，載入解決方案檔案。若要產生開發套件程式庫，請還原 NuGet 套件並建置解決方案。
3. 將 Realtime 用戶端程式庫新增到您的遊戲用戶端專案。

Amazon GameLift ServersRealtime 用戶端 API (C#) 參考：動作

此 C# Realtime用戶端 API 參考可協助您準備多玩家遊戲，以便與Amazon GameLift ServersRealtime 部署在機Amazon GameLift Servers群上的 搭配使用。

- 同步動作
- [非同步回呼](#)
- [資料類型](#)

Client()

初始化新用戶端以與 Realtime 伺服器通訊，並識別要使用的連線類型。

語法

```
public Client(ClientConfiguration configuration)
```

參數

clientConfiguration

指定用戶端/伺服器連線類型的組態詳細資訊。您可以選擇不使用此參數呼叫 Client ()；不過，此方法預設會產生不安全的連線。

類型：[ClientConfiguration](#)

必要：否

傳回值

傳回 Realtime 用戶端執行個體，以與 Realtime 伺服器通訊搭配使用。

Connect()

請求對託管遊戲工作階段的伺服器程序進行連線。

語法

```
public ConnectionStatus Connect(string endpoint, int remoteTcpPort, int listenPort,
    ConnectionToken token)
```

參數

端點

要連接之遊戲工作階段的 DNS 名稱或 IP 地址。端點是在 `GameSession` 物件中指定，該物件會回應用戶端對 AWS SDK Amazon GameLift Servers API 動作 [StartGameSessionPlacement](#)、[CreateGameSession](#) 或 [DescribeGameSessions](#) 的呼叫而傳回。

Note

如果 Realtime 伺服器在具有 TLS 憑證的機群上執行，您必須使用 DNS 名稱。

類型：字串

必要：是

`remoteTcpPort`

指派給遊戲工作階段的 TCP 連線的連接埠號碼。此資訊的指定位置位於 `GameSession` 物件，系統會傳回此物件以回應 [StartGameSessionPlacement](#)、[CreateGameSession](#) 或 [DescribeGameSession](#) 請求。

類型：整數

有效值：1900 到 2000。

必要：是

`listenPort`

遊戲用戶端監聽使用 UDP 管道傳送之訊息所在的連接埠號碼。

類型：整數

有效值：33400 到 33500。

必要：是

`token`

可向伺服器程序識別請求的遊戲用戶端的選用資訊。

類型：[ConnectionToken](#)

必要：是

傳回值

傳回 [ConnectionStatus](#) 列舉值，指出客戶端的連接狀態。

Disconnect()

連接到遊戲工作階段時，將遊戲用戶端與遊戲工作階段中斷連線。

語法

```
public void Disconnect()
```

參數

此動作沒有參數。

傳回值

此方法不會傳回任何內容。

NewMessage()

使用指定的操作程式碼建立新訊息物件。一旦傳回訊息物件，請透過指定目標、更新交付方法，並視需要新增資料承載來完成訊息內容。一旦完成，使用 `SendMessage()` 傳送訊息。

語法

```
public RTMessage NewMessage(int opCode)
```

參數

opCode

開發人員定義的操作程式碼，可識別遊戲事件或動作，例如玩家移動或伺服器通知。

類型：整數

必要：是

傳回值

傳回包含指定操作程式碼和預設交付方法的 [RTMessage](#) 物件。依預設，交付意圖參數是設為 FAST。

SendMessage()

使用指定的交付方法來將訊息傳送給玩家或群組。

語法

```
public void SendMessage(RTMessage message)
```

參數

message

訊息物件，該物件會指定目標收件人、交付方法和訊息內容。

類型：[RTMessage](#)

必要：是

傳回值

此方法不會傳回任何內容。

JoinGroup()

將玩家新增到指定群組的成員資格。群組可以包含連接到遊戲的任何玩家。一旦加入，玩家會收到傳送給群組的所有未來訊息，並能將訊息傳送給整個群組。

語法

```
public void JoinGroup(int targetGroup)
```

參數

targetGroup

可識別要將玩家新增至其中的群組唯一 ID。群組 ID 是由開發人員定義。

類型：整數

必要：是

傳回值

此方法不會傳回任何內容。由於此請求的傳送方式是使用可靠 (TCP) 交付方法，失敗的請求會觸發回呼 [OnError\(\)](#)。

LeaveGroup()

從指定群組的成員資格移除玩家。一旦離開群組，玩家即不再收到系統傳送給群組的訊息，並且無法將訊息傳送給整個群組。

語法

```
public void LeaveGroup(int targetGroup)
```

參數

targetGroup

可識別要從中移除玩家的群組唯一 ID。群組 ID 是由開發人員定義。

類型：整數

必要：是

傳回值

此方法不會傳回任何內容。由於此請求的傳送方式是使用可靠 (TCP) 交付方法，失敗的請求會觸發回呼 [OnError\(\)](#)。

RequestGroupMembership()

請求指定群組中要傳送到遊戲用戶端的玩家清單。任何玩家都可以請求此資訊，不論他們是否為該群組的成員。為了回應此請求，系統會透過 [OnGroupMembershipUpdated\(\)](#) 回呼將成員資格清單傳送到用戶端。

語法

```
public void RequestGroupMembership(int targetGroup)
```

參數

targetGroup

可識別要取得成員資格資訊之群組的唯一 ID。群組 ID 是由開發人員定義。

類型：整數

必要：是

傳回值

此方法不會傳回任何內容。

Amazon GameLift ServersRealtime 用戶端 API (C#) 參考：非同步回呼

使用此 C# Realtime 用戶端 API 參考協助您準備多玩家遊戲，以搭配 Amazon GameLift ServersRealtime 部署在機 Amazon GameLift Servers 群上的使用。

- [同步動作](#)
- 非同步回呼
- [資料類型](#)

遊戲用戶端需要實作這些回呼方法以回應事件。Realtime 伺服器會叫用這些回呼，以將遊戲相關資訊傳送給遊戲用戶端。您也可以在此 Realtime 伺服器指令碼中使用自訂遊戲邏輯來實作相同事件的回呼。請參閱 [的指令碼回呼 Amazon GameLift ServersRealtime](#)。

回呼方法的定義位於 ClientEvents.cs。

OnOpen()

當伺服器程序接受遊戲用戶端的連線請求時叫用，並開啟連線。

語法

```
public void OnOpen()
```

參數

此方法沒有參數。

傳回值

此方法不會傳回任何內容。

OnClose()

當伺服器程序終止與遊戲用戶端的連線時叫用，例如，在遊戲工作階段結束之後。

語法

```
public void OnClose()
```

參數

此方法沒有參數。

傳回值

此方法不會傳回任何內容。

OnError()

當即時用戶端 API 請求發生失敗時叫用。您可以自訂此回呼以處理各種連線錯誤。

語法

```
private void OnError(byte[] args)
```

參數

此方法沒有參數。

傳回值

此方法不會傳回任何內容。

OnDataReceived()

當遊戲用戶端從 Realtime 伺服器接收到訊息時叫用。這是遊戲用戶端接收到訊息和通知的主要方法。

語法

```
public void OnDataReceived(DataReceivedEventArgs dataReceivedEventArgs)
```

參數

dataReceivedEventArgs

訊息活動的相關資訊。

類型：[DataReceivedEventArgs](#)

必要：是

傳回值

此方法不會傳回任何內容。

OnGroupMembershipUpdated()

當玩家所屬群組的成員資格更新時叫用。當用戶端呼叫 `RequestGroupMembership` 時也會叫用此回呼。

語法

```
public void OnGroupMembershipUpdated(GroupMembershipEventArgs groupMembershipEventArgs)
```

參數

groupMembershipEventArgs

群組成員資格活動的相關資訊。

類型：[GroupMembershipEventArgs](#)

必要：是

傳回值

此方法不會傳回任何內容。

Amazon GameLift ServersRealtime 用戶端 API (C#) 參考：資料類型

此 C# Realtime用戶端 API 參考可協助您準備多玩家遊戲，以便與Amazon GameLift ServersRealtime 部署在機Amazon GameLift Servers群上的 搭配使用。

- [同步動作](#)
- [非同步回呼](#)
- 資料類型

ClientConfiguration

遊戲用戶端如何連接到 Realtime 伺服器的資訊。

目錄

ConnectionType

要使用的用戶端/伺服器連線類型 (安全或不安全)。如果您不指定連線類型，預設值是不安全的。

Note

使用 TLS 憑證連接到安全機群上的 Realtime 伺服器時，必須使用 RT_OVER_WSS_DTLS_TLS12 值。

類型：ConnectionType [列舉](#)值。

必要：否

ConnectionToken

請求與 Realtime 伺服器連線的遊戲用戶端和/或玩家的相關資訊。

目錄

playerSessionId

建立新玩家工作階段Amazon GameLift Servers時由發出的唯一 ID。物件中會指定玩家工作階段 ID，此 PlayerSession ID 會回應用戶端對 GameLift API 動作 [StartGameSessionPlacement](#)、[CreateGameSession](#)、[DescribeGameSessionPlacement](#) 或 [DescribePlayerSessions](#) 的呼叫而傳回。

類型：字串

必要：是

payload

要在連線時與 Realtime 伺服器通訊的開發人員定義資訊。這包括可能用於自訂登入機制的任何資料。例如，承載可能會提供要由 Realtime 伺服器指令碼處理之後再允許用戶端連線的身分驗證資訊。

類型：位元組陣列

必要：否

RTMessage

訊息的內容和交付資訊。訊息必須指定目標玩家或目標群組。

目錄

opCode

開發人員定義的操作程式碼，可識別遊戲事件或動作，例如玩家移動或伺服器通知。訊息的操作程式碼可為要提供的資料承載提供內容。使用 `NewMessage()` 建立的訊息已設定操作程式碼，但可以隨時變更。

類型：整數

必要：是

targetPlayer

可識別哪些為要傳送之訊息的預期收件人之玩家的唯一 ID。目標可能是伺服器本身 (使用伺服器 ID) 或其他玩家 (使用玩家 ID)。

類型：整數

必要：否

targetGroup

可識別哪些為要傳送的訊息之預期收件人之群組的唯一 ID。群組 ID 是由開發人員定義。

類型：整數

必要：否

deliveryIntent

指出使用可靠的 TCP 連接或使用快速 UDP 管道來傳送訊息。使用 [NewMessage\(\)](#) 建立的訊息。

類型：DeliveryIntent enum

有效值：FAST | RELIABLE

必要：是

payload

訊息內容。系統會視需要將此資訊結構化，供遊戲用戶端根據伴隨而來的操作程式碼處理使用。它可能包含在遊戲用戶端之間，或遊戲用戶端與 Realtime 伺服器之間通訊所需的遊戲狀態資料或其他資訊。

類型：位元組陣列

必要：否

DataReceivedEventArgs

使用 [OnDataReceived\(\)](#) 回呼提供的資料。

目錄

寄件者

識別產生訊息的實體 (玩家 ID 或伺服器 ID) 的唯一 ID。

類型：整數

必要：是

opCode

開發人員定義的操作程式碼，可識別遊戲事件或動作，例如玩家移動或伺服器通知。訊息的操作程式碼可為要提供的資料承載提供內容。

類型：整數

必要：是

資料

訊息內容。系統會視需要將此資訊結構化，供遊戲用戶端根據伴隨而來的操作程式碼處理使用。它可能包含在遊戲用戶端之間，或遊戲用戶端與 Realtime 伺服器之間通訊所需的遊戲狀態資料或其他資訊。

類型：位元組陣列

必要：否

GroupMembershipEventArgs

使用 [OnGroupMembershipUpdated\(\)](#) 回呼提供的資料。

目錄

寄件者

識別請求群組成員資格更新之玩家的唯一 ID。

類型：整數

必要：是

opCode

可識別遊戲事件或動作的開發人員定義操作程式碼。

類型：整數

必要：是

groupId

可識別哪些為要傳送的訊息之預期收件人之群組的唯一 ID。群組 ID 是由開發人員定義。

類型：整數

必要：是

playerId

指定的群組目前成員的玩家 ID 清單。

類型：Integer array

必要：是

列舉

為的用戶端 SDK 定義的列舉Amazon GameLift ServersRealtime定義如下：

ConnectionStatus

- CONNECTED – 遊戲用戶端僅透過 TCP 連線連線至Realtime伺服器。無論交付意圖為何，所有訊息的傳送都是透過 TCP。
- CONNECTED_SEND_FAST – 遊戲用戶端使用 TCP 和 UDP 連線連線至Realtime伺服器。不過，尚未驗證透過 UDP 接收訊息的能力；因此，傳送給遊戲用戶端的所有訊息都會使用 TCP。
- CONNECTED_SEND_AND_RECEIVE_FAST – 遊戲用戶端使用 TCP 和 UDP 連線連線至Realtime伺服器。遊戲用戶端可以使用 TCP 或 UDP 傳送和接收訊息。
- CONNECTING 遊戲用戶端已傳送連線請求，並且 Realtime 伺服器正在處理請求。
- DISCONNECTED_CLIENT_CALL – 遊戲用戶端已中斷與Realtime伺服器的連線，以回應來自遊戲用戶端的[Disconnect\(\)](#)請求。
- DISCONNECTED – 遊戲用戶端因用戶端中斷通話以外的原因而與Realtime伺服器中斷連線。

ConnectionType

- RT_OVER_WSS_DTLS_TLS12 – 安全連線類型。

用於在產生 TLS 憑證的機Amazon GameLift Servers群上執行的 Realtime 伺服器。使用安全連線時，會使用 TLS 1.2 來加密 TCP 流量，並使用 DTLS 1.2 來加密 UDP 流量。

- RT_OVER_WS_UDP_UNSECURED – 不安全的連線類型。
- RT_OVER_WEBSOCKET – 不安全的連線類型。不再偏好使用此值。

DeliveryIntent

- FAST – 使用 UDP 頻道傳送。
- 可靠 – 使用 TCP 連線傳送。

的指令碼參考 Amazon GameLift ServersRealtime

使用這些資源來在您的 Realtime 指令碼中建置自訂邏輯。

主題

- [的指令碼回呼 Amazon GameLift ServersRealtime](#)
- [Amazon GameLift ServersRealtime 介面](#)

的指令碼回呼 Amazon GameLift ServersRealtime

您可以提供自訂邏輯以回應事件，方法是在 Realtime 指令碼中實作這些回呼。

初始化

初始化 Realtime 伺服器並接收 Realtime 伺服器界面。

語法

```
init(rtsession)
```

onMessage

當收到的訊息傳送到伺服器時叫用。

語法

```
onMessage(gameMessage)
```

onHealthCheck

呼叫以設定遊戲工作階段運作狀態的狀態。根據預設，運作狀態為良好 (或 true)。您可以實作此回呼來執行自訂運作狀態檢查，並傳回狀態。

語法

```
onHealthCheck()
```

onStartGameSession

在新的遊戲工作階段啟動時呼叫，並將遊戲工作階段物件傳遞至其中。

語法

```
onStartGameSession(session)
```

onProcessTerminate

在 Amazon GameLift Servers 服務終止伺服器程序時呼叫。這可做為從遊戲工作階段正常結束的觸發。您不需要呼叫 `processEnding()`。

語法

```
onProcessTerminate()
```

onPlayerConnect

當玩家請求連線並且已通過最初驗證時叫用。

語法

```
onPlayerConnect(connectMessage)
```

onPlayerAccepted

當接受玩家連線時叫用。

語法

```
onPlayerAccepted(player)
```

onPlayerDisconnect

當玩家透過傳送中斷連線請求或其他方式與遊戲工作階段中斷連線時叫用。

語法

```
onPlayerDisconnect(peerId)
```

onProcessStarted

當伺服器處理程序開始時叫用。此回呼允許指令碼執行準備託管遊戲工作階段所需的任何自訂任務。

語法

```
onProcessStarted(args)
```

onSendToPlayer

在伺服器上收到要傳送到另一個玩家的玩家訊息時叫用。此程序會在訊息傳遞之前執行。

語法

```
onSendToPlayer(gameMessage)
```

onSendToGroup

在伺服器上收到要傳送給一個群組的玩家訊息時叫用。此程序會在訊息傳遞之前執行。

語法

```
onSendToGroup(gameMessage))
```

onPlayerJoinGroup

當玩家傳送加入群組的請求時叫用。

語法

```
onPlayerJoinGroup(groupId, peerId)
```

onPlayerLeaveGroup

當玩家傳送離開群組的請求時叫用。

語法

```
onPlayerLeaveGroup(groupId, peerId)
```

Amazon GameLift ServersRealtime 介面

當Amazon GameLift ServersRealtime指令碼初始化時，會傳回Realtime伺服器的介面。本主題會說明透過介面可使用的屬性和方法。在 [建立Realtime指令碼](#) 中進一步了解撰寫 Realtime 指令碼及檢視詳細的指令碼範例。

Realtime 介面提供存取下列物件：

- 工作階段
- player

- gameMessage
- 組態

Realtime 工作階段物件

使用這些方法來存取伺服器相關的資訊，及執行伺服器相關的動作。

getPlayers()

擷取目前連線到遊戲工作階段的玩家對等端 ID 清單。傳回玩家物件的陣列。

語法

```
rtSession.getPlayers()
```

broadcastGroupMembershipUpdate()

觸發將更新後的群組成員資格清單交付至玩家群組。指定要公開的成員資格 (groupIdToBroadcast) 以及要接收更新的群組 (targetGroupId)。群組 IDs 必須是正整數或 "-1"，以表示所有群組。[Amazon GameLift Servers Realtime 指令碼範例](#) 如需使用者定義群組 IDs 的範例，請參閱。

語法

```
rtSession.broadcastGroupMembershipUpdate(groupIdToBroadcast, targetGroupId)
```

getServerId()

擷取伺服器的唯一對等端 ID 識別符，用來將訊息路由至伺服器。

語法

```
rtSession.getServerId()
```

getAllPlayersGroupId()

擷取預設群組的群組 ID，該群組包含所有目前連線至遊戲工作階段的玩家。

語法

```
rtSession.getAllPlayersGroupId()
```

processEnding()

觸發 Realtime 伺服器終止遊戲伺服器。此函數必須從 Realtime 指令碼呼叫，以正常地從遊戲工作階段結束。

語法

```
rtSession.processEnding()
```

getGameSessionId()

擷取目前正在執行的遊戲工作階段唯一 ID。

語法

```
rtSession.getGameSessionId()
```

getLogger()

擷取記錄日誌的界面。使用此方法來記錄將在您遊戲工作階段日誌中擷取的陳述式。記錄器支援使用 "info"、"warn" 和 "error" 陳述式。例如：`logger.info("<string>")`。

語法

```
rtSession.getLogger()
```

sendMessage()

使用 UDP 管道，從 Realtime 伺服器將使用 `newTextGameMessage` 或 `newBinaryGameMessage` 建立的訊息傳送至玩家收件人。使用玩家的對等端 ID 識別收件人。

語法

```
rtSession.sendMessage(gameMessage, targetPlayer)
```

sendGroupMessage()

使用 UDP 管道，從 Realtime 伺服器將使用 `newTextGameMessage` 或 `newBinaryGameMessage` 建立的訊息傳送至玩家群組中所有的玩家。群組 IDs 必須是正整數或 "-1"，以表示所有群組。[Amazon GameLift Servers Realtime 指令碼範例](#) 如需使用者定義群組 IDs 的範例，請參閱。

語法

```
rtSession.sendMessage(gameMessage, targetGroup)
```

sendReliableMessage()

使用 TCP 管道，從 Realtime 伺服器將使用 `newTextGameMessage` 或 `newBinaryGameMessage` 建立的訊息傳送至玩家收件人。使用玩家的對等端 ID 識別收件人。

語法

```
rtSession.sendReliableMessage(gameMessage, targetPlayer)
```

sendReliableGroupMessage()

使用 TCP 管道，從 Realtime 伺服器將使用 `newTextGameMessage` 或 `newBinaryGameMessage` 建立的訊息傳送至玩家群組中所有的玩家。群組 IDs 必須是正整數或 "-1"，以表示所有群組。[Amazon GameLift Servers Realtime 指令碼範例](#) 如需使用者定義群組 IDs 的範例，請參閱。

語法

```
rtSession.sendReliableGroupMessage(gameMessage, targetGroup)
```

newTextGameMessage()

建立包含文字的新訊息，使用 `SendMessage` 函數從伺服器傳送至玩家收件人。訊息格式類似於適用於的用戶端 SDK 中使用的格式 Realtime (請參閱 [RTMessage](#))。其會傳回 `gameMessage` 物件。

語法

```
rtSession.newTextGameMessage(opcode, sender, payload)
```

newBinaryGameMessage()

建立包含二進位資料的新訊息，使用 `SendMessage` 函數從伺服器傳送至玩家收件人。訊息格式類似於適用於的用戶端 SDK 中使用的格式 Realtime (請參閱 [RTMessage](#))。其會傳回 `gameMessage` 物件。

語法

```
rtSession.newBinaryGameMessage(opcode, sender, binaryPayload)
```

玩家物件

存取玩家相關資訊。

`player.peerId`

連線到 Realtime 伺服器並加入遊戲工作階段時，指派給遊戲用戶端的唯一 ID。

`player.playerSessionId`

連線到 Realtime 伺服器並加入遊戲工作階段時，遊戲用戶端參考的玩家工作階段 ID。

遊戲訊息物件

使用這些方法來存取 Realtime 伺服器接收到的訊息。從遊戲用戶端接收到的訊息具有 [RTMessage](#) 結構。

`getPayloadAsText()`

以文字形式取得遊戲訊息承載。

語法

```
gameMessage.getPayloadAsText()
```

`gameMessage.opcode`

訊息中包含的作業碼。

`gameMessage.payload`

訊息中包含的承載。可能是文字或二進位。

`gameMessage.sender`

傳送訊息的遊戲用戶端對等端 ID。

`gameMessage.reliable`

布林值，指出訊息是透過 TCP (true) 還是 UDP (false) 傳送。

組態物件

組態物件可用來覆寫預設組態。

`configuration.maxPlayers`

RealTimeServers 可接受的最大用戶端/伺服器連線數。

預設值為 32。

`configuration.pingIntervalTime`

伺服器將嘗試將 ping 傳送至所有連線用戶端以確認連線狀態良好的時間間隔，以毫秒為單位。

預設值為 3000 毫秒。

遊戲工作階段置放事件

Amazon GameLift Servers 在處理每個遊戲工作階段置放請求時發出事件。您可以將這些事件發佈至 Amazon SNS 主題，如中所述[設定遊戲工作階段放置的事件通知](#)。這些事件也會以近乎即時的方式傳送至 Amazon CloudWatch Events。

本主題說明遊戲工作階段置放事件的結構，並提供每個事件類型的範例。如需遊戲工作階段置放請求狀態的詳細資訊，請參閱 Amazon GameLift Servers API 參考中的 [GameSessionPlacement](#)。

置放事件語法

事件會以 JSON 物件的形式表示。事件結構符合 CloudWatch Events 模式，具有類似的最上層欄位和服務特定詳細資訊。

最上層欄位包括下列項目（如需詳細資訊，請參閱[事件模式](#)）：

`version`

此欄位一律設定為 0（零）。

`id`

事件的唯一追蹤識別符。

詳細資訊類型

值一律為 GameLift Queue Placement Event。

`source`

值一律為 `aws.gamelift`。

帳戶

用來管理 AWS 的帳戶 Amazon GameLift Servers。

time

事件時間戳記。

region

正在處理置放請求 AWS 的區域。這是使用中的遊戲工作階段佇列所在的區域。

resources

正在處理置放請求之遊戲工作階段佇列的 ARN 值。

PlacementFulfilled

已成功完成置放請求。已啟動新的遊戲工作階段，並已為遊戲工作階段放置請求中列出的每個玩家建立新的玩家工作階段。提供玩家連線資訊。

詳細資訊語法：

placementId

指派給遊戲工作階段置放請求的唯一識別符。

port

新遊戲工作階段的連接埠號碼。

gameSessionArn

新遊戲工作階段的 ARN 識別符。

ipAddress

遊戲工作階段的 IP 地址。

dnsName

指派給執行新遊戲工作階段之執行個體的 DNS 識別符。根據執行遊戲工作階段的執行個體是否啟用 TLS，值格式會有所不同。在啟用 TLS 的機群上連線至遊戲工作階段時，玩家必須使用 DNS 名稱，而不是 IP 地址。

啟用 TLS 的機群：`<unique identifier>.<region identifier>.amazongamelift.com`。

Non-TLS-enabled 的機群：`ec2-<unique identifier>.compute.amazonaws.com`。

`startTime`

時間戳記，指出此請求何時放置在佇列中。

`endTime`

時間戳記，指出何時滿足此請求。

`gameSessionRegion`

AWS 託管遊戲工作階段的機群區域。這對應至 `GameSessionArn` 中的區域字元。

`placedPlayerSessions`

在遊戲工作階段置放請求中為每個玩家建立的玩家工作階段集合。

範例

```
{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {
    "type": "PlacementFulfilled",
    "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
    "port": "6262",
    "gameSessionArn": "arn:aws:gamelift:us-west-2::gamesession/fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa/4444dddd-55ee-66ff-77aa-8888bbbb99cc",
    "ipAddress": "98.987.98.987",
    "dnsName": "ec2-12-345-67-890.us-west-2.compute.amazonaws.com",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z",
    "gameSessionRegion": "us-west-2",
  }
}
```

```
    "placedPlayerSessions": [  
      {  
        "playerId": "player-1"  
        "playerSessionId": "psess-1232131232324124123123"  
      }  
    ]  
  }  
}
```

PlacementCancelled

已透過呼叫 GameLift 服務 [StopGameSessionPlacement](#) 來取消置放請求。

詳細資訊：

placementId

指派給遊戲工作階段置放請求的唯一識別符。

startTime

時間戳記，指出此請求何時放置在佇列中。

endTime

時間戳記，指出此請求取消的時間。

範例

```
{  
  "version": "0",  
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",  
  "detail-type": "GameLift Queue Placement Event",  
  "source": "aws.gamelift",  
  "account": "123456789012",  
  "time": "2021-03-01T15:50:52Z",  
  "region": "us-east-1",  
  "resources": [  
    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"  
  ],  
  "detail": {  
    "type": "PlacementCancelled",
```

```
"placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
"startTime": "2021-03-01T15:50:49.741Z",
"endTime": "2021-03-01T15:50:52.084Z"
}
```

PlacementTimedOut

遊戲工作階段置放未在佇列的時間限制過期之前成功完成。可視需要重新提交置放請求。

詳細資訊：

placementId

指派給遊戲工作階段置放請求的唯一識別符。

startTime

時間戳記，指出此請求何時放置在佇列中。

endTime

時間戳記，指出此請求取消的時間。

範例

```
{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {
    "type": "PlacementTimedOut",
    "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z"
  }
}
```

```
}  
}
```

PlacementFailed

Amazon GameLift Servers 無法滿足遊戲工作階段請求。這通常是由非預期的內部錯誤所造成。可視需要重新提交置放請求。

詳細資訊：

placementId

指派給遊戲工作階段置放請求的唯一識別符。

startTime

時間戳記，指出此請求何時放置在佇列中。

endTime

時間戳記指出此請求失敗的時間。

範例

```
{  
  "version": "0",  
  "id": "39c978f3-ba46-3f7c-e787-55bfcca1bd31",  
  "detail-type": "GameLift Queue Placement Event",  
  "source": "aws.gamelift",  
  "account": "252386620677",  
  "time": "2021-03-01T15:50:52Z",  
  "region": "us-east-1",  
  "resources": [  
    "arn:aws:gamelift:us-west-2:252386620677:gamesessionqueue/MegaFrogRace-NA"  
  ],  
  "detail": {  
    "type": "PlacementFailed",  
    "placementId": "e4a1119a-39af-45cf-a990-ef150fe0d453",  
    "startTime": "2021-03-01T15:50:49.741Z",  
    "endTime": "2021-03-01T15:50:52.084Z"  
  }  
}
```

Amazon GameLift Servers AMI 版本

下表識別Amazon GameLift Servers用於受管 EC2 託管的最新 Amazon Machine Image AMIs)。如中所述[Amazon GameLift Servers 中的組態與漏洞分析](#)，您必須定期建立新的Amazon GameLift Servers 受管 EC2 機群，以部署最新的 AMI 版本更新。

與適用於 5+ 的伺服器 SDK Amazon GameLift Servers 搭配使用的 AMIs

Amazon GameLift Servers 使用下列 AMIs 來託管與適用於第 5 Amazon GameLift Servers版的伺服器 SDK 整合的遊戲伺服器。

Amazon GameLift Servers 映像	架構	最新的修補程式	基礎 AWS 映像
Amazon Linux 2023 BASE_AMI_LINUX_2023	x86	2025-06-12	Amazon Linux 2023 AMI 2023.7.20 250609.0 x86_64 HVM 核心-6.1 (版本備註)
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_5	x86	2025-06-12	Amazon Linux 2 核心 5.10 AMI 2.0.20250603.0 x86_64 HVM gp2 (版本備註)
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_5	x86	2025-06-12	Windows_Server-2016-English-Full-Base-2025.06.12 (版本備註)
Amazon Linux 2023 BASE_AMI_LINUX_2023_ARM	ARM64	2025-06-12	Amazon Linux 2023 AMI 2023.7.20 250609.0 arm64 HVM 核心-6.1 (版本備註)
Amazon Linux 2 BASE_AMI_LINUX_2_ARM	ARM64	2025-06-12	Amazon Linux 2 LTS Arm64 核心 5.10 AMI 2.0.20250603.0 arm64 HVM gp2 (版本備註)

與適用於 4 的伺服器 SDK Amazon GameLift Servers 搭配使用的 AMIs

Amazon GameLift Servers 使用以下 AMIs 託管與適用於 第 4 Amazon GameLift Servers 版或更早版本的伺服器 SDK 整合的遊戲伺服器。

Amazon GameLift Servers 映像	架構	最新的修補程式	基礎 AWS 映像
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_4	x86	2025-06-12	Amazon Linux 2 AMI 2.0.20250605.0 x86_64 HVM gp2 (版本備註)
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_4	x86	2025-06-12	Windows_Server-2016-English-Full-Base-2025.06.12 (版本備註)

如需詳細資訊，請參閱下列資源：

- [Amazon Linux 2023 版本備註](#)
- [Amazon Linux 2 版本備註](#)
- [AWS Windows AMI 版本歷史記錄](#)
- [2024 年軟體更新服務和 Windows Server Update Services 內容變更的說明](#)

Amazon GameLift Servers 端點和配額

- 如需可用來以程式設計方式與 服務連線的服務端點和信標端點，請參閱 [Amazon GameLift Servers 服務端點](#)。
- 如需每個 AWS 帳戶使用服務資源或操作的 Amazon GameLift Servers 配額，請參閱 [Amazon GameLift Servers 服務配額](#)。如需配額及如何請求提高配額的詳細資訊，請參閱 [AWS 服務配額](#)。您可以使用 Amazon GameLift Servers 主控台請求提高配額。

Amazon GameLift Servers API 限制

本節中的表格列出 Amazon GameLift Servers API 操作的預設速率限制。這些限制有助於維持服務穩定性，並確保公平使用。每個 API 操作都有根據其資源影響和用量模式的特定限制。

根據預設，每個客戶帳戶的調節限制針對爆量請求設定為 500，針對正常流量設定為 500。此外，每個 API 都有自己的預設限制對，如下所述。

API 限制參考

下表列出 Amazon GameLift Servers API 操作的預設速率限制。標記資源層級限流APIs 可能會受到額外的限制，以防止資料庫熱鍵問題。

API 限制參考

API 動作	高載限制	速率限制	資源層級限流	備註
AcceptMatch	100	100	否	限制是每個帳戶。
ClaimGameServer	40	20	否	限制是每個帳戶。
CreateAlias	4	1	否	限制是每個帳戶。
CreateBuild	10	1	否	限制是每個帳戶。
CreateContainerFleet	4	1	否	限制是每個帳戶。
CreateContainerGroupDefinition	10	1	否	限制是每個帳戶。
CreateFleet	4	1	否	限制是每個帳戶。
CreateFleetLocations	4	1	否	限制是每個帳戶。
CreateGameServerGroup	5	0.1	否	限制是每個帳戶。
CreateGameSession	50	50	否	限制是每個帳戶。
CreateGameSessionQueue	4	1	否	限制是每個帳戶。
CreateLocation	4	1	否	限制是每個帳戶。
CreateMatchmakingConfiguration	4	1	否	限制是每個帳戶。

API 動作	高載限制	速率限制	資源層級 限流	備註
CreateMatchmakingRuleSet	4	1	否	限制是每個帳戶。
CreatePlayerSession	50	50	否	限制是每個帳戶。
CreatePlayerSessions	150	150	否	限制是每個帳戶。
CreateScript	10	1	否	限制是每個帳戶。
CreateVpcPeeringAuthorization	1	0.1	否	限制是每個帳戶。
CreateVpcPeeringConnection	1	0.1	否	限制是每個帳戶。
DeleteAlias	4	1	否	限制是每個帳戶。
DeleteBuild	10	1	否	限制是每個帳戶。
DeleteContainerFleet	4	1	否	限制是每個帳戶。
DeleteContainerGroupDefinition	10	1	否	限制是每個帳戶。
DeleteFleet	4	1	否	限制是每個帳戶。
DeleteFleetLocations	4	1	否	限制是每個帳戶。
DeleteGameServerGroup	5	0.1	否	限制是每個帳戶。
DeleteGameSessionQueue	4	1	否	限制是每個帳戶。
DeleteLocation	4	1	否	限制是每個帳戶。
DeleteMatchmakingConfiguration	4	1	否	限制是每個帳戶。

API 動作	高載限制	速率限制	資源層級 限流	備註
DeleteMatchmakingRuleSet	4	1	否	限制是每個帳戶。
DeleteScalingPolicy	4	1	否	限制是每個帳戶。
DeleteScript	10	1	否	限制是每個帳戶。
DeleteVpcPeeringAuthorization	1	0.1	否	限制是每個帳戶。
DeleteVpcPeeringConnection	1	0.1	否	限制是每個帳戶。
DeregisterGameServer	40	20	否	限制是每個帳戶。
DeregisterCompute	10	2	否	限制是每個帳戶。
DescribeAlias	10	2	否	限制是每個帳戶。
DescribeBuild	20	5	否	限制是每個帳戶。
DescribeCompute	10	2	否	限制是每個帳戶。
DescribeContainerFleet	20	5	否	限制是每個帳戶。
DescribeContainerGroupDefinition	20	5	否	限制是每個帳戶。
DescribeEC2InstanceLimits	40	15	否	限制是每個帳戶。
DescribeFleetAttributes	10	2	否	限制是每個帳戶。

API 動作	高載限制	速率限制	資源層級 限流	備註
DescribeFleetCapacity	4	1	否	限制是每個帳戶。
DescribeFleetDeployment	20	5	否	限制是每個帳戶。
DescribeFleetEvents	10	5	否	限制是每個帳戶。
DescribeFleetLocationAttributes	10	2	否	限制是每個帳戶。
DescribeFleetLocationCapacity	10	5	否	限制是每個帳戶。
DescribeFleetLocationUtilization	10	2	否	限制是每個帳戶。
DescribeFleetPortSettings	10	2	否	限制是每個帳戶。
DescribeFleetUtilization	10	2	否	限制是每個帳戶。
DescribeGameServer	40	20	否	限制是每個帳戶。
DescribeGameServerGroup	10	2	否	限制是每個帳戶。
DescribeGameServerInstances	10	2	否	限制是每個帳戶。
DescribeGameSessionDetails	10	5	否	限制是每個帳戶。請考慮改用 DescribeGameSessions，因為它具有較高的預設限制，並支援資源層級限制。

API 動作	高載限制	速率限制	資源層級 限流	備註
DescribeGameSessionPlacement	75	75	是	限制是每個帳戶。受限於資源層級調節。
DescribeGameSessionQueues	10	2	否	限制是每個帳戶。
DescribeGameSessions	20	20	是	限制是每個帳戶。受限於資源層級調節。
DescribeInstances	10	2	否	限制是每個帳戶。
DescribeMatchmaking	75	75	是	限制是每個帳戶。受限於資源層級調節。
DescribeMatchmakingConfigurations	15	7	否	限制是每個帳戶。
DescribeMatchmakingRuleSets	20	10	否	限制是每個帳戶。
DescribePlayerSessions	40	40	是	限制是每個帳戶。受限於資源層級調節。
DescribeRuntimeConfiguration	4	1	否	限制是每個帳戶。
DescribeScalingPolicies	20	20	否	限制是每個帳戶。
DescribeScript	20	5	否	限制是每個帳戶。
DescribeVpcPeeringAuthorizations	1	0.1	否	限制是每個帳戶。
DescribeVpcPeeringConnections	1	0.1	否	限制是每個帳戶。

API 動作	高載限制	速率限制	資源層級 限流	備註
GetComputeAccess	4	1	否	限制是每個帳戶。
GetComputeAuthToken	10	5	否	限制是每個帳戶。
GetGameSessionLogUrl	4	1	否	限制是每個帳戶。
GetInstanceAccess	4	1	否	限制是每個帳戶。
ListAliases	10	2	否	限制是每個帳戶。
ListBuilds	20	5	否	限制是每個帳戶。
ListContainerFleets	20	5	否	限制是每個帳戶。
ListCompute	10	2	否	限制是每個帳戶。
ListContainerGroupDefinitions	20	5	否	限制是每個帳戶。
ListContainerGroupDefinitionVersions	20	5	否	限制是每個帳戶。
ListFleets	10	2	否	限制是每個帳戶。
ListFleetDeployments	20	5	否	限制是每個帳戶。
ListGameServerGroups	10	2	否	限制是每個帳戶。
ListGameServers	10	2	否	限制是每個帳戶。
ListLocations	10	2	否	限制是每個帳戶。
ListScripts	20	5	否	限制是每個帳戶。
ListTagsForResource	20	5	否	限制是每個帳戶。

API 動作	高載限制	速率限制	資源層級 限流	備註
PutScalingPolicy	4	1	否	限制是每個帳戶。
RegisterCompute	10	5	否	限制是每個帳戶。
RegisterGameServer	40	20	否	限制是每個帳戶。
RequestUploadCredentials	4	1	否	限制是每個帳戶。
ResolveAlias	4	1	否	限制是每個帳戶。
ResumeGameServerGroup	1	0.1	否	限制是每個帳戶。
SearchGameSessions	10	5	否	限制是每個帳戶。
StartFleetActions	1	0.1	否	限制是每個帳戶。
StartGameSessionPlacement	75	75	否	限制是每個帳戶。
StartMatchBackfill	70	70	否	限制是每個帳戶。
StartMatchmaking	100	100	否	限制是每個帳戶。
StopFleetActions	1	0.1	否	限制是每個帳戶。
StopGameSessionPlacement	20	20	否	限制是每個帳戶。
StopMatchmaking	100	100	否	限制是每個帳戶。
SuspendGameServerGroup	1	0.1	否	限制是每個帳戶。
TagResource	20	5	否	限制是每個帳戶。

API 動作	高載限制	速率限制	資源層級 限流	備註
TerminateGameSession	10	2	否	限制是每個帳戶。
UntagResource	20	5	否	限制是每個帳戶。
UpdateAlias	4	1	否	限制是每個帳戶。
UpdateBuild	10	1	否	限制是每個帳戶。
UpdateContainerGroupDefinition	10	1	否	限制是每個帳戶。
UpdateContainerFleet	4	1	否	限制是每個帳戶。
UpdateFleetAttributes	4	1	否	限制是每個帳戶。
UpdateFleetCapacity	4	1	否	限制是每個帳戶。
UpdateFleetPortSettings	4	1	否	限制是每個帳戶。
UpdateGameServer	40	20	否	限制是每個帳戶。
UpdateGameServerGroup	5	0.1	否	限制是每個帳戶。
UpdateGameSession	10	2	否	限制是每個帳戶。
UpdateGameSessionQueue	4	1	否	限制是每個帳戶。
UpdateMatchmakingConfiguration	4	1	否	限制是每個帳戶。
UpdateRuntimeConfiguration	4	1	否	限制是每個帳戶。
UpdateScript	10	1	否	限制是每個帳戶。

API 動作	高載限制	速率限制	資源層級限流	備註
ValidateMatchmakingRuleSet	4	1	否	限制是每個帳戶。

資源層級限流

有些 API 操作會受到資源層級限流，以防止資料庫熱鍵問題。在提高這些操作的 API 層級限制時，它們仍可能在資源層級受到限流。

下列 API 操作受到資源層級限流的限制：

- DescribeGameSessionPlacement
- DescribeGameSessions
- DescribeMatchmaking
- DescribePlayerSessions

使用包含資源識別符的特定限流金鑰強制執行資源層級限流。例如，DescribeGameSessionPlacement 使用金鑰 "Operation : GameLift/DescribeGameSessionPlacement，aws-account-placement-id："，可強制執行每個帳戶和置放 ID 組合的限流。這可防止客戶對特定資源提出請求，讓系統不堪負荷。

Amazon GameLift Servers SDK API 限制

下表列出 Amazon GameLift Servers SDK API 操作的預設速率限制。這些限制可最佳化效能，並防止遊戲伺服器環境中的資源爭用。了解這些限制對於與 Amazon GameLift 伺服器進行有效的伺服器端整合至關重要。

伺服器 SDK 限制會使用包含帳戶和程序識別符的特定限流金鑰強制執行。有些限制是每個玩家，以防止個別玩家濫用。

Note

此資料表可能會有所變更。目前限制的授權來源是內部 SDC 組態。如有需要，請聯絡 AWS Support 以取得 up-to-date。

伺服器開發套件 API 限制參考

API 動作	高載限制	速率限制	調節金鑰	備註
AcceptPlayerSession	3	1	aws-account-and-process-id-and-player-id	限制是每個玩家和每個程序。不應該需要提高限制，因為它是每個玩家。
RemovePlayerSession	3	1	aws-account-and-process-id-and-player-id	限制是每個玩家和每個程序。不應該需要提高限制，因為它是每個玩家。
ActivateGameSession	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
ActivateHostProcessV2	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
StartMatchBackfill	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
DescribePlayerSessions	10	5	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
GetComputeCertificate	3	0.1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
GetCustomRoleCredentials	3	0.1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。

API 動作	高載限制	速率限制	調節金鑰	備註
InitSDK	3	0.1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
RefreshHostProcess	3	0.1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
StopMatchmaking	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
ProcessEnding	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
ProcessReady	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
UpdateGameSession	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
UpdateHostProcess	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
GetGameSessionId	10	5	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。
GetTerminationTime	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。

API 動作	高載限制	速率限制	調節金鑰	備註
UpdatePlayerSessionCreationPolicy	3	1	aws-account-and-process-id	限制是每個伺服器程序。永遠不需要提高限制，因為它是每個程序。

Note

這些是預設限制。如果您在這些 APIs 上遇到限流，可能是因為帳戶層級限流。若要請求提高限制，請聯絡 AWS Support。

每個程序和每個玩家的限制旨在適應正常的使用模式，並且很少需要調整。如果您遇到這些 APIs 限流的情況，請在請求提高限制之前，先檢閱您的實作是否有潛在的最佳化機會。

帳戶層級限流也可能透過下列其他規則套用：

- SdkWebSocket/AccountLevelHighUsage，aws-account
- SdkWebSocket/AccountLevelThrottle，aws-account

Amazon GameLift Servers UDP ping 信標

UDP ping 信標提供一種方法來測量玩家裝置和 Amazon GameLift Servers 託管位置之間的網路延遲。使用 ping 信標，您可以收集準確的延遲資料，以做出有關遊戲伺服器置放的明智決策，並根據延遲需求改善玩家配對。

UDP ping 信標的運作方式

Amazon GameLift Servers 在您可以部署遊戲伺服器的每個託管位置提供固定的 UDP 端點 (ping 信標)。由於大多數遊戲伺服器使用 UDP 進行通訊，因此使用 UDP ping 信標測量延遲可提供比使用 ICMP ping 更準確的結果。網路裝置處理 ICMP 封包的方式通常與 UDP 封包不同，這可能會導致延遲測量，無法反映玩家將體驗到的實際效能。

使用 UDP ping 信標，您的遊戲用戶端可以傳送 UDP 訊息到這些端點並接收非同步回應，為您提供延遲測量，更能代表玩家裝置與潛在託管位置之間的實際遊戲流量條件。端點是永久的，只要 Amazon GameLift Servers 支援在該位置託管遊戲，端點就會保持可用。

UDP ping 信標的常見使用案例

您可以透過多種方式使用 UDP ping 信標來最佳化遊戲的聯網體驗。

選擇最佳託管位置

跨不同地理區域收集延遲資料，以識別託管玩家基礎遊戲伺服器的最佳主要和備份位置。

根據玩家延遲放置遊戲工作階段

在請求新的遊戲工作階段時包含玩家延遲資料，以協助挑選可提供最低延遲體驗的位置。

根據延遲最佳化配對

在請求配對時提供玩家延遲資料，以協助配對具有類似延遲描述檔的玩家，並將遊戲工作階段放置在配對玩家的最佳位置。

Note

建立配對請求時，您不應將延遲資訊提供給沒有機群的位置。如果您這麼做，Amazon GameLift Servers 可能會嘗試將遊戲工作階段放置在沒有機群容量的位置，導致配對請求失敗。

取得信標端點

若要擷取 Amazon GameLift Servers 位置的 ping 信標網域和連接埠資訊，請使用 [ListLocations](#) API 操作。此 API 傳回的一組位置取決於 AWS 區域您在呼叫它時指定的（或者，如果您未指定預設區域）。當您從呼叫時：

- 支援多位置機群的主區域：API 會傳回所有託管位置的資訊
- 支援單一位置之機群的主區域：API 會傳回該位置的資訊

請注意，如果您使用只能是多位置機群中遠端位置的位置呼叫此 API，API 將傳回錯誤，因為該類型的位置沒有服務端點。

請參閱 [中支援的位置表格](#) [支援 AWS 的位置](#)，以識別支援單一和多位置機群的主區域。

範例

```
aws gamelift list-locations --region ap-northeast-2
```

這 AWS 區域 支援多位置機群，因此會傳回多個位置。以下是其中一個傳回值的範例：

```
[...]  
{  
  "LocationName": "ap-northeast-1",  
  "PingBeacon": {  
    "UDPEndpoint": {  
      "Domain": "gamelift-ping.ap-northeast-1.api.aws",  
      "Port": 7770  
    }  
  }  
}
```

Important

快取 ping 信標資訊，而不是在每次延遲測量ListLocations之前呼叫。網域和連接埠資訊是靜態的，API 並非專為大量請求而設計。

實作延遲測量

使用 UDP ping 信標實作延遲測量時，請遵循下列最佳實務：

1. 使用下列其中一種方法來存放 ping 信標資訊：
 - 硬式編碼遊戲用戶端中的端點。
 - 快取遊戲後端中的資訊。
 - 實作定期更新機制（每日/每週）以重新整理資訊。
2. 傳送 UDP ping 訊息：
 - 只要訊息內文不是空的，且訊息的大小上限為 300 位元組，即可將您想要的任何內容放入訊息內文。
 - 請遵守每個位置的下列速率限制：
 - 每個唯一寄件者 IP 地址和連接埠組合每秒 3 次交易 (TPS)
 - 每個唯一寄件者 IP 地址 1000 TPS
3. 計算延遲：
 - 將多個 Ping 傳送至每個位置，以計算平均延遲。
 - 考慮將並行 ping 傳送到多個位置，以提高結果。

- 視需要使用重試邏輯，為未在短時間內傳回的任何封包傳送新封包（通常是 1 - 3 秒），因為 UDP 沒有 100% 保證交付。
- 計算傳送訊息和接收回應之間的時間差異。
- 如果對某個位置的大部分 UDP ping 始終無法傳回回應，請使用 ICMP ping 計算標準[Amazon GameLift Servers服務端點](#)的延遲作為備用。

Tip

無論您在何處記錄玩家在其本機網路上必須開啟的連接埠清單，都建議您包含連接埠 7770。這是另一個原因，因為您應該有後援來測量延遲（例如使用 ICMP），以防此連接埠遭到封鎖。

程式碼範例

以下是一些簡單的範例，示範如何傳送 UDP ping 和計算延遲。

C++

```
#include <iostream>
#include <cstring>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <chrono>

int main() {
    // Replace with Amazon GameLift Servers UDP ping beacon domain for your desired
    location
    const char* domain = "gamelift-ping.ap-south-1.api.aws";
    const int port = 7770;
    const char* message = "Ping";           // Your message
    const int num_pings = 3;                // Number of pings to send

    // Create socket
    int sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock < 0) {
        std::cerr << "Error creating socket" << std::endl;
        return 1;
    }
```

```
}

// Resolve domain name to IP address
struct hostent* host = gethostbyname(domain);
if (host == nullptr) {
    std::cerr << "Error resolving hostname" << std::endl;
    close(sock);
    return 1;
}

// Set up the server address structure
struct sockaddr_in server_addr;
std::memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
std::memcpy(&server_addr.sin_addr, host->h_addr, host->h_length);

// Set socket timeout
struct timeval tv;
tv.tv_sec = 1; // 1 second timeout
tv.tv_usec = 0;
if (setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv)) < 0) {
    std::cerr << "Error setting socket timeout" << std::endl;
    close(sock);
    return 1;
}

double total_latency = 0;
int successful_pings = 0;

for (int i = 0; i < num_pings; ++i) {
    auto start = std::chrono::high_resolution_clock::now();

    // Send the message
    ssize_t bytes_sent = sendto(sock, message, std::strlen(message), 0,
                                (struct sockaddr*)&server_addr,
sizeof(server_addr));

    if (bytes_sent < 0) {
        std::cerr << "Error sending message" << std::endl;
        continue;
    }

    // Receive response
```



```

    char buffer[1024];
    socklen_t server_addr_len = sizeof(server_addr);
    ssize_t bytes_received = recvfrom(sock, buffer, sizeof(buffer), 0,
                                      (struct sockaddr*)&server_addr,
    &server_addr_len);

    auto end = std::chrono::high_resolution_clock::now();

    if (bytes_received < 0) {
        std::cerr << "Error receiving response or timeout" << std::endl;
    } else {
        std::chrono::duration<double, std::milli> latency = end - start;
        total_latency += latency.count();
        successful_pings++;

        std::cout << "Received response, latency: " << latency.count() << " ms"
    << std::endl;
    }

    // Wait a bit before next ping
    usleep(1000000); // 1s
}

// Close the socket
close(sock);

if (successful_pings > 0) {
    double avg_latency = total_latency / successful_pings;
    std::cout << "Average latency: " << avg_latency << " ms" << std::endl;
} else {
    std::cout << "No successful pings" << std::endl;
}

return 0;
}

```

C#

```

using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Diagnostics;

```

```
using System.Threading.Tasks;

class UdpLatencyTest
{
    static async Task Main()
    {
        // Replace with Amazon GameLift Servers UDP ping beacon domain for your
        // desired location
        string domain = "gamelift-ping.ap-south-1.api.aws";
        int port = 7770;
        string message = "Ping";           // Your message
        int numPings = 3;                  // Number of pings to send
        int timeoutMs = 1000;              // Timeout in milliseconds

        await MeasureLatency(domain, port, message, numPings, timeoutMs);
    }

    static async Task MeasureLatency(string domain, int port, string message, int
numPings, int timeoutMs)
    {
        using (var udpClient = new UdpClient())
        {
            try
            {
                // Resolve domain name to IP address
                IPAddress[] addresses = await Dns.GetHostAddressesAsync(domain);
                if (addresses.Length == 0)
                {
                    Console.WriteLine("Could not resolve domain name.");
                    return;
                }

                IPEndPoint endPoint = new IPEndPoint(addresses[0], port);
                byte[] messageBytes = Encoding.UTF8.GetBytes(message);

                // Set receive timeout
                udpClient.Client.ReceiveTimeout = timeoutMs;

                double totalLatency = 0;
                int successfulPings = 0;
                var stopwatch = new Stopwatch();

                for (int i = 0; i < numPings; i++)
                {
```

```
        try
        {
            stopwatch.Restart();

            // Send message
            await udpClient.SendAsync(messageBytes, messageBytes.Length,
endPoint);

            // Wait for response
            UdpReceiveResult result = await
ReceiveWithTimeoutAsync(udpClient, timeoutMs);

            stopwatch.Stop();
            double latency = stopwatch.Elapsed.TotalMilliseconds;

            totalLatency += latency;
            successfulPings++;

            string response = Encoding.UTF8.GetString(result.Buffer);
            Console.WriteLine($"Ping {i + 1}: {latency:F2}ms - Response:
{response}");
        }
        catch (SocketException ex)
        {
            Console.WriteLine($"Ping {i + 1}: Failed - {ex.Message}");
        }
        catch (TimeoutException)
        {
            Console.WriteLine($"Ping {i + 1}: Timeout");
        }

        // Wait before next ping
        await Task.Delay(1000); // 1s between pings
    }

    if (successfulPings > 0)
    {
        double averageLatency = totalLatency / successfulPings;
        Console.WriteLine($"\\nSummary:");
        Console.WriteLine($"Successful pings: {successfulPings}/
{numPings}");
        Console.WriteLine($"Average latency: {averageLatency:F2}ms");
    }
    else
```

```

        {
            Console.WriteLine("\nNo successful pings");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}

static async Task<UdpReceiveResult> ReceiveWithTimeoutAsync(UdpClient client,
int timeoutMs)
{
    using var cts = new System.Threading.CancellationTokenSource(timeoutMs);
    try
    {
        return await client.ReceiveAsync().WaitAsync(cts.Token);
    }
    catch (OperationCanceledException)
    {
        throw new TimeoutException("Receive operation timed out");
    }
}
}

```

Python

```

import socket
import time
import statistics
from datetime import datetime

def udp_ping(host, port, timeout=2):
    """
    Send a UDP ping and return the round trip time in milliseconds.
    Returns None if timeout occurs.
    """
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.settimeout(timeout)
    message = b'ping'

    try:

```

```
# Resolve hostname first
try:
    socket.gethostbyname(host)
except socket.gaierror as e:
    print(f"Could not resolve hostname: {e}")
    return None

start_time = time.time()
sock.sendto(message, (host, port))

# Wait for response
data, server = sock.recvfrom(1024)
end_time = time.time()

# Calculate round trip time in milliseconds
rtt = (end_time - start_time) * 1000
return rtt

except socket.timeout:
    print(f"Request timed out")
    return None
except Exception as e:
    print(f"Error: {type(e).__name__}: {e}")
    return None
finally:
    sock.close()

def main():
    # Replace with Amazon GameLift Servers UDP ping beacon domain for your desired
    location
    host = "gamelift-ping.ap-south-1.api.aws"
    port = 7770
    num_pings = 3
    latencies = []

    print(f"\nPinging {host}:{port} {num_pings} times...")
    print(f"Start time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")

    for i in range(num_pings):
        print(f"Ping {i+1}:")
        rtt = udp_ping(host, port)

        if rtt is not None:
            print(f"Response from {host}: time={rtt:.2f}ms")
```

```
latencies.append(rtt)

# Wait 1 second between pings
if i < num_pings - 1:
    time.sleep(1)
print()

# Calculate and display statistics
print("-" * 50)
print(f"Ping statistics for {host}:")
print(f"    Packets: Sent = {num_pings}, Received = {len(latencies)}, "
      f"Lost = {num_pings - len(latencies)} "
      f"f"{{{(num_pings - len(latencies)) / num_pings * 100}:.1f}% loss}")

if latencies:
    print("\nRound-trip latency statistics:")
    print(f"    Minimum = {min(latencies):.2f}ms")
    print(f"    Maximum = {max(latencies):.2f}ms")
    print(f"    Average = {statistics.mean(latencies):.2f}ms")

print(f"\nEnd time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")

if __name__ == "__main__":
    main()
```

Amazon GameLift Servers 版本備註提供與服務相關的新功能、更新及修正的詳細資訊。

下表列出具有 SDK 版本資訊的所有 Amazon GameLift Servers 版本。您的遊戲伺服器 and 用戶端整合不需要使用類似的 SDKs。不過，某個 SDK 的較早版本可能無法完全支援另一個 SDK 的最新功能。

Amazon GameLift Servers SDKs和外掛程式是開放原始碼。請參閱 [取得Amazon GameLift Servers開發工具](#)。若要取得最新版本，請參閱 [Amazon GameLift Servers GitHub 組織](#)。

目前版本

[illegible]

舊版本

 Unreal Engine

 引擎

 引擎

 引擎

 引擎DK

 式

 Unity

 +

 不

 真

 實

 Unreal Engine

 或

 更

 新

 反

 本

 Unreal Engine

 或

 更

 新

 反

 本

 Unreal Engine

 .NET

 更

 新

 反

 本

Realtime
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

Realtime
Unity
不
真
實

170
或
更
新
反
本

151
或
更
新
反
本

151
或
更
新
反
本

Realtime
Gameplay
Script
Engine
DK
式

Unity
+
不
真
實

[5.2.12.0.251](#)
或
更
新
反
本

[5.2.12.0.251](#)
或
更
新
反
本

[5.2.12.0.251](#)
或
更
新
反
本

式

實

本

本

本

Realtime
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

[Script](#)
Script
Script
Script
Script
Script
Script

[Script](#)
Script
Script
Script
Script
Script
Script

[Script](#)
Script
Script
Script
Script
Script
Script

Realtime	
Overview	
Architecture	
Deployment	
Configuration	
DK	
Style	
%	
Unity	
+	
不	
真	
實	

502406-1	
或	
更	
新	
反	
本	

502405-2	
或	
更	
新	
反	
本	

502404-2	
或	
更	
新	
反	
本	

Realtime
Gameplay
Script
Engine
DK
式

Unity
+
不
真
實

[500004-1](#)
或
更
新
反
本

[500002-0](#)
或
更
新
反
本

[500001-3](#)
或
更
新
反
本

Run Time

藏用

聯和

業

糖DK

式



Unity

+

不

真

實

~~5142010~~ 5142012-0

NET

E.NET

ॐ

反

本

~~314220~~B-2

或

更

折

反

本

~~3012200~~-2

或

更

折

反

本

Realtime
Script
Server
SDK
Style

Unity
+
不
真
實

[3.0.1206-0](#)
或
更
新
反
本

[3.0.1208-2](#)
或
更
新
反
本

[3.0.1208-1](#)
或
更
新
反
本

Realtime
Script
Server
SDK
Style
Unity
+
不
真
實

[3.0.130-0](#)
或
更
新
反
本

[3.0.130-2](#)
或
更
新
反
本

[3.0.120-2](#)
或
更
新
反
本

Realtime
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

[Script 201-1](#)
Script
Script
Script
Script
Script
Script

[Script 209-1](#)
Script
Script
Script
Script
Script
Script

[Script 208-2](#)
Script
Script
Script
Script
Script
Script

式

實

本

本

本

Realtime
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

式

實

本

本

本

Realtime
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

[B33012-0](#)
Script
Script
Script
Script
Script

[B33012-1](#)
Script
Script
Script
Script
Script

[B22109-2](#)
Script
Script
Script
Script
Script

式

實

本

本

本

Realtime
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

[B.2.202-0](#)
Script
Script
Script
Script
Script

[B.1.1709-0](#)
Script
Script
Script
Script
Script

[B.1.1708-1](#)
Script
Script
Script
Script
Script

Realtime
物理
圖形
聲卡
網路
模擬
物理DK
程式
式

Unity
+
不
真
實

[BC175025-1](#)

或
更
新
反
本

[BC175024-1](#)

或
更
新
反
本

[BC175022-2](#)

或
更
新
反
本

Realtime
Script
Script
Script
Script
Script
Script

Script
Script
Script
Script
Script
Script
Script

[Script 1-1](#)
Script
Script
Script
Script
Script

[Script 10-1](#)
Script
Script
Script
Script
Script

[Script 19-0](#)
Script
Script
Script
Script
Script

- 封裝 Android 用戶端目標解決時的編譯問題。

進一步了解：

- [Unreal Engine 的 Amazon GameLift Servers 外掛程式整合遊戲](#)，Amazon GameLift Servers 開發人員指南
- [整合 Amazon GameLift Servers 到 Unreal Engine 專案](#)，Amazon GameLift Servers 開發人員指南

2025 年 6 月 24 日：Amazon GameLift Servers 啟動 UDP ping 信標，以協助測量多玩家遊戲的即時網路延遲

已更新 SDK 版本：

- AWS 開發套件 1.11.595

Amazon GameLift Servers 發行一般可用性的一組固定端點，稱為 UDP ping 信標，可協助您準確測量玩家裝置和遊戲伺服器位置之間的延遲。UDP ping 信標端點可在支援的所有 AWS 全球區域和本地區域使用 Amazon GameLift Servers，但 AWS 中國區域除外。

大多數多玩家遊戲使用 UDP（使用者資料包通訊協定）作為其主要封包傳輸通訊協定，因為其對即時遊戲的效能優勢。了解和最佳化網路延遲對於提供最佳玩家體驗至關重要。UDP ping 信標提供一致且可靠的方法來測量玩家和遊戲伺服器之間的實際 UDP 封包延遲，協助您針對 player-to-server 配對和遊戲工作階段置放做出更好的決策。

[ListLocations](#) API 已擴充，將端點網域和連接埠資訊包含在傳回的位置清單中，以程式設計方式輕鬆存取端點。

您的遊戲用戶端可以傳送 UDP 訊息到這些端點，並使用相同的資料接收非同步回應，為您提供延遲測量，更能代表玩家裝置和潛在託管位置之間的實際遊戲流量條件。這些端點是永久的，只要 Amazon GameLift Servers 支援在該位置託管遊戲，這些端點就會保持可用。

進一步了解：

- [UDP ping 信標](#)、Amazon GameLift Servers 開發人員指南
- [ListLocations](#)、Amazon GameLift Servers API 參考

2025 年 6 月 24 日：Amazon GameLift Servers 新增泰國和馬來西亞的位置支援

使用 Amazon GameLift Servers 受管託管，您現在可以在泰國曼谷和馬來西亞吉隆坡部署遊戲伺服器資源，將遊戲的觸角擴展到東南亞的玩家。這些新區域有助於降低延遲並改善這些區域中玩家的遊戲體驗。

以下是多位置機群的 AWS 區域 遠端位置。若要在這些位置開始託管遊戲工作階段，請將它們新增為遠端位置到新的或現有的多位置機群。使用多位置機群，您可以直接管理每個位置的託管容量。

- 亞太區域（泰國）(ap-southeast-7)
- 亞太區域（馬來西亞）(ap-southeast-5)

AWS 帳戶預設 AWS 區域 不會啟用這些項目。您必須先將選擇加入每個區域，才能在該處部署 Amazon GameLift Servers 資源。

進一步了解：

- [Amazon GameLift Servers 服務位置](#)、Amazon GameLift Servers 託管開發人員指南
- [更新機群位置](#)、Amazon GameLift Servers 託管開發人員指南
- [啟用或停用您帳戶中 AWS 的區域](#)，AWS 帳戶管理參考指南。

2025 年 5 月 29 日：針對 Unreal Engine 和 Unity Amazon GameLift Servers 發行新版本的伺服器 SDK 和外掛程式

已更新 SDK 版本：

- [C++ 伺服器 SDK](#)，5.3.0 版
- [C# 伺服器 SDK](#)，5.3.0 版
- [Go 伺服器 SDK](#)，5.3.0 版
- [適用於 Unreal 的伺服器 SDK](#)，5.3.0 版
- [適用於 Unity 的伺服器 SDK](#)，5.3.0 版

更新外掛程式版本：

- [Unreal 的外掛程式](#)，3.0.0 版
- [Unity 的外掛程式](#)，3.1.0 版

適用於 C++、C#、Go、Unreal 和 Unity 的新版本伺服器 SDK，以及適用於 Unreal Engine 和 Unity 的新外掛程式版本，現在都是開放原始碼。它們都可以在 [Amazon GameLift Servers GitHub 組織](#) 中使用。如需詳細更新資訊，請參閱每個儲存庫中的版本備註和讀寫。

金鑰伺服器 SDK 更新：

- 改善所有伺服器 SDKs 中的用戶端驗證和錯誤回應。
- [OnProcessTerminate](#) 回呼函數現在具有預設邏輯來結束遊戲伺服器程序。
- 函數 [InitSDK\(\)](#) 現在使用冪等字符來支援多次重試。
- [OnUpdateGameSession](#) 回呼現在可以傳遞 `autoBackfillMode` 的資料。

金鑰外掛程式更新：

- Unreal Engine 的外掛程式現在具有更簡化的安裝和設定程序，具有更多自動化和更少的先決條件 (CMake、OpenSSL 和 Unreal 跨編譯工具鏈)。
- Unreal Engine 的外掛程式改善了受管 EC2 工作流程的 UI 體驗，包括對遊戲用戶端和伺服器建置路徑中空間的支援。此外，您現在可以在從編輯器啟動遊戲用戶端時新增命令列引數。
- Unreal Engine 的外掛程式現在支援 UE5 中的 ARM 伺服器建置。

進一步了解：

- [Amazon GameLift Servers Unreal Engine 的外掛程式](#)、Amazon GameLift Servers 開發人員指南
- [Amazon GameLift Servers Unity 的外掛程式 \(伺服器 SDK 5.x\)](#)、Amazon GameLift Servers 開發人員指南
- [適用於的伺服器 SDK 5.x Amazon GameLift Servers](#)、Amazon GameLift Servers 開發人員指南
- [Amazon GameLift Servers 整合至 Unreal Engine 專案](#)、Amazon GameLift Servers 開發人員指南
- [Amazon GameLift Servers 整合至 Unity 專案](#)、Amazon GameLift Servers 開發人員指南

2025 年 4 月 24 日：適用於的 C# 伺服器 SDK 5 Amazon GameLift Servers 現在支援 .NET 8

已更新 SDK 版本：

- C# 伺服器 SDK，5.2.1 版

對於使用 C# 的遊戲開發人員，您現在可以使用 .NET 8 做為 Amazon GameLift Servers 專案的目標架構。透過 .NET 8，您可以利用效能改善，包括改善 just-in-time (JIT) 編譯、記憶體使用量最佳化和更快的啟動時間。如果您目前正在使用 .NET 6，我們建議您規劃遷移至 .NET 8，包括將 C# 伺服器 SDK 更新至最新版本。Microsoft 宣布 .NET 8 支援安全修補程式和技術更新，截至 2026 年 11 月。

[下載適用於 的最新版本 C# 伺服器開發套件 Amazon GameLift Servers](#)

進一步了解：

- [取得 Amazon GameLift Servers 開發工具](#)、Amazon GameLift Servers 開發人員指南

2025 年 3 月 27 日：Amazon GameLift Servers 擴展對新一代 EC2 執行個體系列的支援

已更新 SDK 版本：

- AWS SDK 1.11.535

您現在可以使用 從第 5 到第 8 代執行個體系列中的更多 Amazon EC2 執行個體選擇 Amazon GameLift Servers 進行微調。每一代新的 EC2 都提供 EC2 運算、記憶體和聯網的進展，而第 8 代執行個體則提供尖端的 AWS Graviton4 和 Intel Xeon 型執行個體。新一代執行個體可在下列執行個體系列中使用：

- [一般用途](#) (M 系列)
- [運算最佳化](#) (C 系列)
- [記憶體最佳化](#) (R 系列)

您也可以選擇提供本機儲存 (d)、增強型聯網 (n) 和特定處理器架構 (Intel/AMD/Graviton – i/a/g) 的變體。除了 AWS 中國區域以外 Amazon GameLift Servers，新一代執行個體可在 AWS 區域 支援的所有中使用。如需詳細資訊，請參閱 [Amazon GameLift Servers 服務位置](#)。

將這些新的執行個體類型與 Amazon GameLift Servers 受管 EC2 機群和受管容器機群搭配使用。將現有遊戲託管切換到新的執行個體類型（相同架構）時，只需部署新的機群，所有組態設定都保持不變，但執行個體類型除外。

進一步了解：

- [Amazon GameLift Servers 執行個體定價](#)
- [CreateFleet](#) Amazon GameLift Servers API 參考

- [CreateContainerFleet](#)、Amazon GameLift ServersAPI 參考

2025 年 3 月 11 日：Amazon GameLift Servers 推出新的 Game Server Wrapper 以進行快速入門

的新遊戲伺服器包裝函式可Amazon GameLift Servers大幅縮短取得託管於 的遊戲伺服器所需的時間 Amazon GameLift Servers。無需變更程式碼，您可以使用包裝函式將基本遊戲工作階段管理功能新增至遊戲，並將其部署到 Amazon GameLift Servers Anywhere 機群、受管 EC2 機群或受管容器機群。此工具非常適合使用您自己的遊戲伺服器或範例遊戲中的伺服器，對Amazon GameLift Servers功能進行實作評估。它也適用於快速部署遊戲伺服器反覆運算，例如快速原型設計或測試。

透過基本遊戲工作階段管理功能，您的遊戲伺服器可以初始化與服務的連線Amazon GameLift Servers、回應啟動和停止遊戲工作階段的提示，以及在遊戲工作階段完成時關閉。

[從 GitHub 下載遊戲伺服器包裝函式](#)。如需如何搭配遊戲專案安裝和使用包裝函式，請參閱 GitHub 上的讀我檔案。

2025 年 3 月 6 日：Amazon GameLift 現在和 Amazon GameLift Servers Amazon GameLift Streams

Amazon GameLift 是一項全受管服務，專門協助開發人員建置、擴展和交付全球最嚴苛的遊戲。透過 Amazon GameLift Streams 的一般可用性版本，Amazon GameLift 現在同時提供高規模的遊戲伺服器和順暢的遊戲串流功能。

[Amazon GameLift Servers](#) 讓遊戲開發人員能夠部署、操作和擴展專用遊戲伺服器。您可以在幾分鐘內在雲端部署高效能遊戲伺服器，並擴展和縮減以滿足玩家需求。以 AWS 經過驗證的運算環境為基礎，在單一遊戲中Amazon GameLift Servers支援 1 億個並行玩家、每秒新增 10 萬個玩家，以及每分鐘 9 千個新的運算執行個體。此外，透過企業級安全性、最大群眾的配對，以及pay-as-you-go彈性，無論您正在處理新的遊戲想法，或是執行有數百萬名玩家的遊戲，都可協助您開始使用。

[Amazon GameLift Streams](#) 可協助遊戲開發人員以高達 1080p 的解析度和frames-per-second (fps) 提供遊戲串流體驗，而不會在包括 iOS、Android、FireOS 和適用於玩家PCs 等裝置上產生延遲。使用單一 AWS 產品，發佈者可以在幾分鐘內將其遊戲內容部署到完全受管的雲端 GPU 執行個體上，無需修改，並透過 AWS 網路骨幹將它們直接交付到具有 Web 瀏覽器的任何最終玩家裝置。玩家可以在幾秒鐘內開始遊戲，無需等待下載或安裝，而且可提供幾乎與在 PC 或遊戲主控台上本機遊戲無關的遊戲體驗。

2025 年 1 月 14 日：Amazon GameLift Servers 發佈遊戲工作階段置放優先順序的擴展支援

已更新 SDK 版本：

- AWS SDK 1.11.485

為了回應客戶意見回饋，我們發佈了新功能，可讓您優先處理個別遊戲工作階段置放請求的位置。對於已設定為依位置排定置放優先順序的佇列，您現在可以為每個置放請求提供自訂的優先順序清單。

這項新功能可讓客戶視需要動態變更每個置放請求的位置優先順序。額外的彈性意味著您可以更好地回應不斷變化的條件，例如玩家位置、機群負載或伺服器運作狀態。它也可以支援想要進一步自訂放置位置選取方式的客戶。

進一步了解：

- [排定遊戲工作階段置放的優先順序](#)，Amazon GameLift Servers開發人員指南
- [StartGameSessionPlacement](#)，Amazon GameLift ServersAPI 參考

2025 年 1 月 2 日：Amazon GameLift Servers 發佈終止遊戲工作階段的新支援

已更新 SDK 版本：

- AWS SDK 1.11.477

為了回應客戶意見回饋，我們正在發佈新功能，讓您更輕鬆地終止個別遊戲工作階段。透過此版本，您現在可以直接在Amazon GameLift Servers主控台或使用適用於的 AWS CLI 或 AWS SDK 來終止遊戲工作階段Amazon GameLift Servers。

這項新功能解決了解決仍處於作用中但狀態不良的遊戲工作階段的需求，這可防止運算資源託管新的遊戲工作階段。先前，客戶需要遠端存取運算，才能手動終止遊戲工作階段。

您有兩種終止方法可供選擇。第一個方法會嘗試使用其自訂關機順序來正常終止遊戲工作階段，其中可能包括通知玩家並解析遊戲資料的動作。第二個方法強制伺服器程序停止，這會立即終止遊戲工作階段。第二個方法可確保即使伺服器程序沒有回應，遊戲工作階段也會結束。

進一步了解：

- [使用 Amazon GameLift Servers主控台關閉遊戲工作階段](#)，Amazon GameLift Servers開發人員指南

- [TerminateGameSession](#)、Amazon GameLift ServersAPI 參考

2024 年 12 月 19 日：Amazon GameLift Servers 發行受管容器的遊戲引擎外掛程式支援

已更新外掛程式版本：

Amazon GameLift ServersUnreal Engine 的外掛程式，2.0.0 版

- 升級為支援具有受管容器支援的 C++ 伺服器 SDK 5.2.0。
- 新增對 Unreal Engine 5.4 和 5.5 的支援。

Amazon GameLift ServersUnity 的外掛程式，3.0.0 版

- 升級為支援具有受管容器支援的 C++ 伺服器 SDK 5.2.0。
- 支援適用於 Windows 和 Mac OS 的 Unity 2021.3 LTS 和 2022.3 LTS。

Unreal 和 Unity 遊戲引擎的 Amazon GameLift Servers 外掛程式提供工具和工作流程，可簡化您使用啟動和執行遊戲的步驟Amazon GameLift Servers。Amazon GameLift Servers 是一種全受管雲端託管服務，遊戲開發人員可用來管理和擴展工作階段型多玩家遊戲的專用遊戲伺服器。

最新的外掛程式版本提供下列增強功能：

- 使用 受管容器託管的引導工作流程。此工作流程會逐步引導您使用遊戲伺服器軟體設定容器映像，並為您的遊戲伺服器部署雲端託管解決方案。工作流程提供兩種不同的部署案例：簡單的部署，以及具有遊戲工作階段置放佇列和FlexMatch配對建構器的更完整部署。每個案例都會產生Amazon GameLift Servers容器機群和支援 AWS 的資源。
- 改善設定 AWS 使用者設定檔和管理外掛程式使用 AWS 存取憑證的程序。您可以維護多個設定檔，以使用不同的 AWS 帳戶、帳戶使用者和區域。
- 更新現有容器機群的其他功能。您可以部署新的容器映像（例如遊戲伺服器版本更新）和變更機群組態設定，而不必從頭開始。
- 改善了使用 Amazon GameLift Servers Anywhere 機群和受管 EC2 託管的工作流程。根據客戶意見回饋的改進包括更好的指引，其中包含實用資源的秘訣和連結。

Managed Containers 和 Managed EC2 解決方案的部署案例使用 AWS CloudFormation 範本來建立和部署每個案例 AWS 的資源。這些範本包含在 Amazon GameLift Servers 外掛程式下載中且可編輯。您可以照原樣使用它們，或為您的遊戲修改它們。

進一步了解：

- [Unreal 的外掛程式：將您的遊戲部署到受管容器機群](#)，Amazon GameLift Servers 開發人員指南
- [Unity 外掛程式：將您的遊戲部署到受管容器機群](#)，Amazon GameLift Servers 開發人員指南
- [從 GitHub 下載適用於 Unreal 的外掛程式](#)
- [從 GitHub 下載 Unity 的外掛程式](#)
- [使用 託管遊戲 Amazon GameLift Servers](#)
- [Amazon GameLift Servers 論壇](#)

2024 年 11 月 12 日：使用受管容器 Amazon GameLift Servers 啟動多玩家遊戲託管

已更新 SDK 版本：

- AWS SDK 1.11.445
- 伺服器 SDK，5.2.0 版（所有語言）

Amazon GameLift Servers 針對一般可用性的版本是容器化遊戲伺服器工作負載的新託管解決方案。在此版本中，遊戲開發人員現在可以利用容器化的優勢，包括一致、安全的環境、簡化的部署程序和最佳化的資源使用率。

受管容器機群會使用 Amazon GameLift Servers 代您管理的 Amazon EC2 執行個體，並根據您的組態進行管理。您可以為遊戲建置自訂容器架構，並將容器映像儲存在 Amazon Elastic Container Registry (Amazon ECR) 儲存庫中，以提供容器映像。容器機群僅適用於 Linux 型遊戲伺服器。遊戲伺服器必須與伺服器 SDK 5.2.0 或更新版本整合。

透過受管容器機群，您可以獲得與受管 EC2 機群相同的優勢。這包括支援隨需和 Spot 執行個體類型、智慧容量擴展、使用佇列放置遊戲工作階段，以及配對。您也可以取得與其他機群類型相同的指標，以及容器的一些新指標。容器機群的其他功能包括：

- 與容器化工作負載的無伺服器體驗保持一致。每個容器執行一個遊戲伺服器程序，並將許多容器封裝到每個機群執行個體上，以獲得最佳資源使用量。如果您偏好使用具有多個遊戲伺服器程序的容器，您可以使用 Amazon GameLift Servers 代理程式進行自動主機管理。

- 簡化機群建立。容器機群的設計需要最少的部署組態設定，以及合理的建議/預設值。您可以快速部署工作機群，然後視需要自訂個別設定。
- 容器架構的版本控制工具。您現在可以更新容器群組定義（類似於容器「任務」）、維護多個版本，並指定要部署到機群的版本。
- 機群更新工具。使用容器機群時，當您想要發行遊戲伺服器版本更新時，不再需要建立新的機群。相反地，您現在可以更新容器映像，並將更新部署到現有的機群。

您可以在 AWS 區域 Amazon GameLift Servers 支援多位置機群的任何 中建置 Amazon GameLift Servers 容器機群，也可以將容器機群執行個體部署到任何支援的遠端位置。如需詳細資訊，請參閱 [Amazon GameLift Servers 服務位置](#)。受管容器目前無法在 AWS 在中國區域使用。

進一步了解：

- 部落格文章：[利用全受管容器在 上以全球規模託管多玩家遊戲 Amazon GameLift Servers](#)
- [受管容器](#) 概觀、Amazon GameLift Servers 開發人員指南
- [容器在 中的運作方式 Amazon GameLift Servers](#)、Amazon GameLift Servers 開發人員指南
- [使用 Amazon GameLift Servers 受管容器託管的開發藍圖](#)、Amazon GameLift Servers 開發人員指南
- [CreateContainerFleet](#)、Amazon GameLift Servers API 參考

2024 年 9 月 19 日：Amazon GameLift Servers 發行更新至 Unreal Engine 的 C++ 伺服器 SDK 和外掛程式

已更新 SDK 版本：

C++ 伺服器 SDK，5.1.3 版

- 新的記錄功能。您現在可以存取 SDK 請求日誌。
- 改善 SDK 訊息傳輸可靠性。開發套件現在使用更強大的重新連線機制，在網路中斷或隨機訊息中斷時進行復原。

更新外掛程式版本：

Amazon GameLift Servers Unreal Engine 的外掛程式，1.1.2 版

- 升級以支援最新版本的 C++ 伺服器 SDK 5.1.3。

- 在 Unreal Engine 的 Amazon GameLift Servers 外掛程式中，當瀏覽機群的伺服器建置可執行檔時，您現在可以選擇瀏覽所有檔案。

適用於 Unreal 的 C++ 伺服器 SDK 外掛程式，5.1.2 版

- 升級以支援最新版本的 C++ 伺服器 SDK 5.1.3。

進一步了解：

- [Unreal Engine 的 Amazon GameLift Servers 外掛程式整合遊戲](#)，Amazon GameLift Servers 開發人員指南
- [Amazon GameLift Servers 外掛程式和軟體開發套件下載](#)

2024 年 9 月 5 日：Amazon GameLift Servers 改善機群建立程序的可觀測性

根據客戶的意見回饋，我們釐清了建立受管 EC2 機群的 Amazon GameLift Servers 工作流程，並準備好託管遊戲工作階段。改良功能包括：

- 我們已針對機群建立程序的每個階段提供更具體且準確的描述。這種改善的可見性可讓您更輕鬆地快速找出和解決問題。
- 建置和啟用階段會將執行個體部署任務（建置）與任務分開，以啟動遊戲伺服器程序並連線至 Amazon GameLift Servers 服務（啟用）。此變更可讓您更輕鬆地識別問題的可能原因。此外，您現在可以在機群處於啟用階段時從遠端連線至機群。
- 兩個新的機群建立事件會傳達遊戲伺服器安裝指令碼的成功或失敗。如果您是遊戲伺服器建置包含安裝指令碼，會 Amazon GameLift Servers 嘗試執行指令碼並發出下列其中一個新事件：
 - FLEET_CREATION_COMPLETED_INSTALLER
 - FLEET_CREATION_FAILED_INSTALLER

進一步了解：

- [Amazon GameLift Servers 機群建立的運作方式](#)、Amazon GameLift Servers 開發人員指南
- [偵錯 Amazon GameLift Servers 機群問題](#)、Amazon GameLift Servers 開發人員指南
- [事件資料類型](#)、Amazon GameLift Servers API 參考

2024 年 8 月 15 日：Amazon GameLift Servers改善主控台體驗

根據客戶的意見回饋，我們已對[Amazon GameLift Servers主控台](#)體驗進行下列更新：

- 頁面的顯示偏好設定現在會自動儲存到 AWS 您的帳戶使用者，並會在您返回頁面時套用。顯示偏好設定可讓您選擇要包含在資料表顯示中的資訊，例如在機群清單頁面上。使用資料表右上角的圖示自訂您的顯示偏好設定。
- 已簡化受管 EC2 機群的建立機群工作流程，以結合機群位置和執行個體類型的選擇。即使您變更位置選擇，我們也能讓您更輕鬆地為您的機群找到正確的執行個體類型。

進一步了解：

- [建立Amazon GameLift Servers受管 EC2 機群](#)、Amazon GameLift Servers 開發人員指南

2024 年 7 月 25 日：Amazon GameLift Servers 新增對 AWS 奈及利亞本地區域的支援

透過Amazon GameLift Servers受管託管，您現在可以在奈及利亞、西非部署遊戲伺服器資源，並將遊戲的範圍擴展到全非洲的玩家。使用 AWS Local Zones 將遊戲伺服器放置在地理位置上更接近玩家的位置，以減少延遲並大幅改善遊戲體驗。

若要立即開始在奈及利亞託管遊戲工作階段，請將新的奈及利亞本地區域做為遠端位置新增至新的或現有的多位置機群。如果您的遊戲使用 Amazon GameLift Servers FlexMatch，請更新配對佇列中的機群，以包含新的 Local Zone。透過多位置機群，您可以直接管理每個位置的託管容量。

奈及利亞 AWS 區域 拉哥斯本地區域的父系是非洲（開普敦）區域 (af-south-1)，Amazon GameLift Servers也支援 做為遠端位置。奈及利亞本地區域名稱為 af-south-1-los-1。

進一步了解：

- [Amazon GameLift Servers 服務位置](#)、Amazon GameLift Servers 開發人員指南
- [更新機群位置](#)、Amazon GameLift Servers 開發人員指南

2024 年 7 月 2 日：Amazon GameLift Servers 發佈新的主控台工具以檢視玩家工作階段資料

Amazon GameLift Servers 主控台現在提供玩家工作階段查詢工具，可讓您依遊戲工作階段 ID、玩家工作階段 ID 或玩家 ID 擷取玩家工作階段資訊。使用 FlexMatch 配對的遊戲會自動為每個配對玩家產生玩家工作階段。對於所有其他遊戲，玩家工作階段是選用功能。

您可以在 Amazon GameLift Servers 主控台的主要導覽中找到玩家工作階段查詢工具。檢視個別玩家工作階段或比較多個玩家工作階段的資料。您也可以在檢視遊戲工作階段詳細資訊頁面時開啟玩家工作階段資料。

進一步了解：

- [Amazon GameLift Servers 主控台中的遊戲和玩家工作階段](#)、Amazon GameLift Servers 開發人員指南

2024 年 4 月 24 日：Amazon GameLift Servers 發行容器機群預覽

Amazon GameLift Servers 現在提供容器機群的預覽，這可讓您改善可攜性、可擴展性、容錯能力和敏捷性。

在容器機群中，Amazon EC2 執行個體託管一或多個容器。這些容器包含遊戲伺服器及其所需的一切，包括相依性和組態。相依性的範例包括 SDKs 和軟體套件。將容器上傳至私有 Amazon Elastic Container Registry 之後，Amazon GameLift Servers 會將容器填入您的機群。

若要在容器機群中運作，您的遊戲伺服器必須在 Linux 中執行，並與伺服器 SDK 5.x 整合。在容器機群中，您可以微調託管資源的控制，以便最佳化 CPU 單位和記憶體等資源的耗用。您也可以在容器中託管多個遊戲伺服器，以減少資源的使用。

在容器機群中，您可以獲得與其他類型機群相同的許多優點，例如隨需執行個體類型、擴展（自動和手動）、佇列和配對。您也可以取得與其他機群類型相同的指標，以及容器的一些新指標。容器機群可讓您全域接觸這些位置區域中的玩家：

- ap-northeast-1
- ap-northeast-2
- ap-southeast-2
- eu-central-1
- eu-west-1
- us-east-1

- us-west-2

若要觸及更多區域和本機區域，請建立多位置容器機群。

進一步了解：

- [使用Amazon GameLift Servers容器管理託管](#)，Amazon GameLift Servers開發人員指南
- [CreateContainerGroupDefinition](#)、Amazon GameLift ServersAPI 參考

2024 年 2 月 13 日：Amazon GameLift Servers 啟動SDKs的改進，並簡化 Unreal Engine Amazon GameLift Servers外掛程式的安裝

已更新 SDK 版本：

- Go Server SDK，5.1.0 版
- C# 伺服器 SDK，5.1.2 版
- C++ 伺服器 SDK，5.1.2 版

我們進行了下列改進：

- 改善 SDK 的可靠性，方法是在網路中斷時新增自動重新連線。
- **【Go】** 您現在可以InitSDK()使用或不使用伺服器參數來呼叫。在Amazon GameLift Servers 受管 EC2 機群上執行的遊戲伺服器會直接從環境變數讀取伺服器參數。Amazon GameLift ServersAnywhere 機群上的遊戲伺服器必須使用InitSDK()伺服器參數呼叫。

已更新外掛程式版本：

- Amazon GameLift ServersUnreal Engine 的外掛程式，1.1.0 版
- Amazon GameLift Servers Unity 的外掛程式，2.1.0 版
- 適用於 Unreal 的 C++ 伺服器 SDK 外掛程式，5.1.1 版
- 適用於 Unity 的 C# 伺服器 SDK 外掛程式，5.1.2 版

我們進行了下列改進：

- **【適用於 Unreal Engine 的Amazon GameLift Servers外掛程式】** 更新了安裝說明並簡化了封裝。此外掛程式現在包含最新版本的 C++ Server SDK for Unreal。

- 升級外掛程式以支援適用於 的最新版本伺服器 SDKAmazon GameLift Servers。

進一步了解：

- [Unreal Engine 的 Amazon GameLift Servers外掛程式整合遊戲](#)，Amazon GameLift Servers開發人員指南
- [Amazon GameLift Servers 外掛程式和軟體開發套件下載](#)

2023 年 12 月 14 日：Amazon GameLift Servers 新增更新作用中遊戲工作階段遊戲屬性的功能

您已經能夠在建立遊戲工作階段時設定遊戲屬性，以及搜尋指定屬性的遊戲工作階段。現在您也可以在使用中的遊戲工作階段中新增和更新這些屬性。

例如，您的玩家在他們想要播放的地圖上投票。您的遊戲用戶端會呼叫 `UpdateGameSession`，將 `GameProperty` 值修改為 `{"Key": "map", "Value": "jungle"}`。然後，您的遊戲會為遊戲工作階段中的玩家實作新的地圖。

遊戲管理員也可以使用 `SearchGameSessions` 操作，從遊戲屬性擷取有用的資料。例如，管理員可以列出 `Status` 值為 `ACTIVE` 以及此遊戲屬性：`{"Key": "map", "Value": "desert"}`。

進一步了解：

- [the section called “Amazon GameLift Servers 新增至遊戲用戶端”](#)、Amazon GameLift Servers 開發人員指南
- [GameProperty](#)，Amazon GameLift ServersAPI 參考
- [UpdateGameSession](#)、Amazon GameLift ServersAPI 參考
- [SearchGameSessions](#)、Amazon GameLift ServersAPI 參考

2023 年 11 月 21 日：Amazon GameLift Servers 啟動對 Infrastructure as Code 工具的支援，例如採用的 Terraform 和 Pulumi AWS 雲端控制 API

您現在可以使用基礎設施即程式碼 (IaC) 工具來管理整個 Amazon GameLift Servers 資源堆疊。這些工具包括 AWS CloudFormation，以及第三方工具，例如 Terraform 和 Pulumi。透過這項新增的支援，您現在可以專注於建置遊戲，並利用 DevOps 策略來負責資源管理、CI/CD 和部署給客戶。

您也可以使用 AWS Cloud Control API 來佈建和設定所有 Amazon GameLift Servers 資源類型。您可以使用 Amazon GameLift Servers APIs 或適用於 的 AWS CloudFormation 範本繼續使用 資源 Amazon GameLift Servers。

如需透過 IaC 可用 Amazon GameLift Servers 資源的詳細資訊，請參閱 [Amazon GameLift Servers 資源類型參考](#) Amazon GameLift Servers 資源類型參考。

此外，您現在可以使用 AWS CloudFormation 範本或 AWS Cloud Control API 自動擴展機群，方法是使用新的 [機群](#) 屬性：ScalingPolicies。

Cloud Control API 為開發人員提供一組標準 APIs，以跨數百個 AWS 服務和多個第三方工具建立、讀取、更新、刪除和列出資源 (CRUDL)，例如 Terraform 和 Pulumi。

進一步了解：

- [AWS CloudFormation](#)
- [AWS 雲端控制 API](#)
- [AWS CC Terraform 提供者](#)
- [Pulumi](#)

2023 年 11 月 16 日：Amazon GameLift Servers 更新 Unity 的獨立外掛程式

已更新 SDK 版本：適用於 Unity 的 Amazon GameLift Servers 外掛程式，版本 2.0.0

Unity 的 Amazon GameLift Servers 外掛程式提供工具和工作流程，簡化使用 啟動和執行 Unity 遊戲以進行雲端託管的步驟 Amazon GameLift Servers。Amazon GameLift Servers 是一種全受管服務，可讓遊戲開發人員管理和擴展工作階段型多玩家遊戲的專用遊戲伺服器。

在此版本中，Unity 的外掛程式已更新為使用最新 Amazon GameLift Servers 功能，包括伺服器 SDK 5.x 版，以及支援 Amazon GameLift Servers Anywhere 進行本機測試。外掛程式與 Unity 版本 Unity 2021.3 LTS 和 2022.3 LTS 相容。

主要外掛程式功能包括：

- Unity 編輯器中的引導式 UI 工作流程適用於下列案例：
 - Amazon GameLift Servers 使用本機工作站做為主機，測試與 的遊戲整合。此工作流程可協助您為本機機器設定 Amazon GameLift Servers Anywhere 機群、啟動遊戲伺服器和用戶端的執行個體、透過 請求遊戲工作階段 Amazon GameLift Servers，以及加入遊戲。

- 使用Amazon GameLift Servers受管 EC2 和支援 AWS 的資源，為您的整合遊戲伺服器部署雲端託管解決方案。此工作流程可協助您為雲端託管設定遊戲，並提供三種部署案例：
 - 將遊戲伺服器部署至單一機群。
 - 將遊戲伺服器部署到多個 AWS 區域中的一組低成本 Spot 機群。
 - 使用 FlexMatch 配對建構器部署遊戲伺服器。
- 能夠設定連結至 AWS 帳戶使用者的使用者設定檔，並設定預設 AWS 區域。您可以維護多個設定檔，以便在不同的 AWS 帳戶、帳戶使用者和區域中運作。
- 協助簡化Amazon GameLift Servers整合和部署程序的特別便利性，包括：
 - 每個託管解決方案都包含支援 AWS 的資源，包括提供唯一玩家 IDs和玩家驗證的 Amazon Cognito 使用者集區。解決方案也包含用於儲存的 Amazon S3 儲存貯體、Amazon SNS 事件通知、AWS Lambda 函數和其他資源。
 - 對於 Anywhere 工作流程，外掛程式會自動執行所需的伺服器參數設定。
 - 對於 Amazon EC2 工作流程，每個部署解決方案都會使用 Lambda 函數提供內建的用戶端後端服務。後端服務位於遊戲用戶端和服務之間，Amazon GameLift Servers並管理對Amazon GameLift Servers服務的所有直接呼叫。
- 整合測試的內容，包括簡單範例多玩家遊戲的資產和程式碼，以說明遊戲伺服器和遊戲用戶端整合。
- 包含詳細整合指引和範本程式碼的外掛程式文件。

包括適用於 Anywhere 和 Amazon EC2 機群的所有部署案例，都使用 AWS CloudFormation 範本來描述和部署遊戲解決方案 AWS 的資源。這些範本包含在Amazon GameLift Servers外掛程式下載中。您可以照原樣使用它們，或為您的遊戲自訂它們。

進一步了解：

- [Amazon GameLift ServersUnity 的外掛程式 \(伺服器 SDK 5.x\)](#)、Amazon GameLift Servers 開發人員指南
- [從 GitHub 下載外掛程式](#)
- [關於Amazon GameLift Servers託管](#)
- [Amazon GameLift Servers 論壇](#)

2023 年 11 月 2 日：Amazon GameLift Servers 新增對共用登入資料的支援

已更新 SDK 版本：AWS SDK 1.11.193

新的Amazon GameLift Servers共用登入資料功能可讓部署在受管 EC2 機群上的應用程式與其他 AWS 資源互動。此更新會影響您綁定和部署的應用程式，以及與伺服器 SDK 5.x 版或更新版本整合的遊戲伺服器二進位檔。（遊戲伺服器可執行檔已經可以使用伺服器 SDK 5.x `GetFleetRoleCredentials()`動作請求登入資料。）

例如，如果您想要使用 Amazon CloudWatch 代理程式部署遊戲伺服器組建以收集 EC2 執行個體指標和其他資料，代理程式需要許可才能與您的 CloudWatch 資源互動。若要這樣做，您必須先設定具有使用 CloudWatch 資源許可的 AWS Identity and Access Management (IAM) 角色，然後設定已啟用 IAM 角色和共用登入資料的機群。當 Amazon GameLift Servers 將遊戲伺服器建置部署到每個 EC2 執行個體時，會產生共用的登入資料檔案，並將其存放在執行個體上。執行個體上的所有應用程式都可以使用共用的登入資料。會在執行個體的整個生命週期中Amazon GameLift Servers自動重新整理暫時登入資料。

您可以使用下列方法，在建立受管 EC2 機群時啟用共用憑證：

- 在Amazon GameLift Servers主控台機群建立工作流程中。
- `CreateFleet` 使用新參數 呼叫服務 API 操作時`InstanceRoleCredentialsProvider`。
- `aws gamelift create-fleet` 使用 參數 呼叫 AWS CLI 操作時`instance-role-credentials-provider`。

進一步了解：

- [與機群中的其他 AWS 資源通訊](#)，Amazon GameLift Servers開發人員指南
- [CreateFleet、InstanceRoleCredentialsProvider](#)、Amazon GameLift ServersAPI 參考
- [設定 IAM 服務角色](#)、Amazon GameLift Servers開發人員指南

2023 年 9 月 28 日：針對 Unreal Engine Amazon GameLift Servers發行新的獨立外掛程式

已更新 SDK 版本：適用於 Unreal Engine 1.0.0 版的Amazon GameLift Servers外掛程式

Unreal Engine 的 Amazon GameLift Servers外掛程式提供工具和工作流程，可簡化啟動遊戲並使用 執行Amazon GameLift Servers雲端託管的步驟。Amazon GameLift Servers 是一種全受管服務，可讓遊戲開發人員管理和擴展工作階段型多玩家遊戲的專用遊戲伺服器。外掛程式支援 UE 5.0、5.1 和 5.2 版。主要特色包括：

- **Unreal 編輯器】** 中的引導 UI 工作流程逐步完成下列路徑：

- Amazon GameLift Servers 使用本機工作站做為主機，測試與 的遊戲整合。此工作流程可協助您為本機機器設定 Amazon GameLift Servers Anywhere 機群、啟動遊戲伺服器 and 用戶端的執行個體、透過 請求遊戲工作階段 Amazon GameLift Servers，以及取得新遊戲工作階段的連線資訊。
- 為您的整合遊戲伺服器部署 Amazon EC2 雲端託管解決方案。此工作流程可協助您為雲端託管設定遊戲，並提供三種不同的部署案例：部署至單一機群、部署至多個區域中的一組 Spot 機群，或使用 FlexMatch 配對建構器部署至一組機群。每個部署案例的解決方案都包含 Amazon GameLift Servers 資源和支援 AWS 的資源。
- 能夠設定連結至 AWS 帳戶使用者的使用者設定檔，並定義預設 AWS 區域。您可以維護多個設定檔，以便在不同的 AWS 帳戶、帳戶使用者和區域中運作。
- 協助簡化 Amazon GameLift Servers 整合和部署程序的特別便利性，包括：
 - 每個託管解決方案都包含支援 AWS 資源，包括提供唯一玩家 IDs 的基本 Amazon Cognito 使用者集區、用於儲存的 Amazon S3 儲存貯體、Amazon SNS 事件通知和 AWS Lambda 函數。
 - 對於 Anywhere 工作流程，外掛程式會使用命令列引數自動化所需的伺服器參數設定。
 - 對於 Amazon EC2 工作流程，每個部署解決方案都會使用 Lambda 函數提供內建的用戶端後端服務。後端服務會收到來自遊戲用戶端的請求，並將其傳遞給 Amazon GameLift Servers 服務。
- 整合測試的內容，包括入門遊戲地圖和兩個具有基本藍圖和 UI 元素的測試地圖。
- 包含詳細整合指引和範本程式碼的外掛程式文件。

所有部署案例，包括適用於 Anywhere 和 Amazon EC2 機群，都使用 AWS CloudFormation 範本來描述解決方案。外掛程式會在為您的遊戲部署 Amazon GameLift Servers 資源時使用這些範本。這些範本包含在 Amazon GameLift Servers 外掛程式下載中且可編輯。您可以照原樣使用它們，或為您的遊戲修改它們。

進一步了解：

- [Amazon GameLift Servers Unreal Engine 的外掛程式](#)、Amazon GameLift Servers 開發人員指南
- [從 GitHub 下載外掛程式](#)
- [關於 Amazon GameLift Servers 託管](#)
- [Amazon GameLift Servers 論壇](#)

2023 年 8 月 17 日：Amazon GameLift Servers 提供使用 AWS Graviton 處理器託管的遊戲伺服器

更新的 SDK 版本：AWS SDK 1.11.144

透過 Amazon GameLift Servers，您現在可以使用 EC2 執行個體搭配 AWS Graviton 處理器在雲端託管遊戲。Graviton 執行個體由 AWS 使用 Arm64-based 處理器設計，可為使用 EC2 的雲端工作負載提供最佳價格效能，比類似的 x86 型執行個體改善高達 40%。相較於舊版，最新的 Graviton3 處理器可提供高達 25% 的運算效能。

使用 Amazon GameLift Servers，您現在可以從 AWS Graviton 系列中的這些新執行個體中選取：

- Graviton2-based 執行個體：c6g、c6gn、r6g、m6g、g5g
- Graviton3-based 執行個體：c7g、r7g、m7g

進一步了解：

- [AWS Graviton 處理器](#)：了解 Graviton EC2 執行個體的優點和實際用途。
- [Graviton 入門](#)：取得 Graviton 型執行個體的概觀，以及應用程式如何根據其作業系統、語言和執行時間對其執行的洞見。

Note

Graviton Arm 執行個體需要在 Linux 作業系統上建置 Amazon GameLift Servers 伺服器。C++ 和 C# 需要伺服器 SDK 5.1.1 或更新版本。Go 需要伺服器 SDK 5.0 或更新版本。這些執行個體在 out-of-the-box 可用支援。AL2023AL2

2023 年 7 月 27 日：Amazon GameLift Servers 發行伺服器 SDK 5.1.0，並新增對 Unity 開發的支援

更新 SDK 版本：適用於 C++、C#/Unity、Unreal 5.1.0 的伺服器 SDK

伺服器 Amazon GameLift Servers 開發套件的最新版本提供 C++、C# 和 Unreal 外掛程式的更新，以及可與 Unity 遊戲引擎搭配使用的新外掛程式。遊戲開發人員將 Amazon GameLift Servers 伺服器 SDK 整合到他們部署用於在 上託管的遊戲伺服器 Amazon GameLift Servers。

最新的伺服器 SDK 版本包含下列更新，其中包括許多客戶請求：

- 下載特定語言 SDK 套件 – 更新的 [Amazon GameLift Servers 下載網站](#) 包含每種語言的 SDK 套件。您可以下載目前或先前的版本。
- 適用於 Unity 的新 C# 伺服器 SDK 外掛程式 – 適用於 Unity 的新伺服器 SDK 套件包含內建的 C# 程式庫，您可以使用 Unity Editor 中的套件管理員來安裝（請參閱新的 [Unity 整合指南](#)）。這些程式庫

透過 UnityNuGet 包含必要的相依性。您可以搭配 Unity 2020.3 LTS、2021.3 LTS 和 2022.3 LTS for Windows 和 Mac OS 使用此外掛程式。它支援 Unity 的 .NET Framework 和 .NET Standard 設定檔，以及 .NET Standard 2.1 和 .NET 4.x。

- 適用於 C# 的合併 .NET 解決方案 – 適用於 C# 的伺服器開發套件現在支援單一解決方案中的 .NET Framework 4.6.2（從 4.6.1 升級）和 .NET 6.0。Unity 建置的程式庫提供 .NET Standard 2.1。
- 伺服器 SDK 5.1.0 更新
 - **【C++、C#、Unreal】** 您現在可以 `InitSDK()` 使用或不使用伺服器參數來呼叫。在 Amazon GameLift Servers 受管 EC2 機群上執行的遊戲伺服器會直接從環境變數讀取伺服器參數。Amazon GameLift Servers Anywhere 機群上的遊戲伺服器必須使用 `InitSDK()` 伺服器參數呼叫。
 - **【C++、C#、Unreal】** 伺服器 SDK 呼叫改善了錯誤訊息。
 - **【C++ SDK】** 為了改善伺服器 SDK 建置時間，預設 `-DRUN_CLANG_FORMAT` 會停用建置旗標。您可以使用 啟用它 `-DRUN_CLANG_FORMAT=1`。
 - **【C++ SDK】** 在沒有標準程式庫 (`-DGAMELIFT_USE_STD=0`) 的情況下建置程式庫時，`InitSDK()` 不再使用 `std::` 資料類型。
- 擴充的伺服器 SDK 5.x 文件
 - 更新 C++、C#/Unity 和 Unreal 的伺服器 SDK 參考指南，包括所有資料類型的擴展涵蓋範圍。
 - [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)
 - [適用於 -- 動作的 C++ 伺服器 SDK Amazon GameLift Servers 5.x](#)
 - [適用於 -- 動作的 C++（非真實）伺服器 SDK Amazon GameLift Servers 5.x](#)
 - 適用於 Unity 和 Unreal 外掛程式的新版伺服器 SDK 5 整合指南
 - [Amazon GameLift Servers 整合至 Unity 專案](#)
 - [Amazon GameLift Servers 整合至 Unreal Engine 專案](#)
- 其他文件更新
 - 已修訂 Amazon GameLift Servers 服務 API 操作 [GetComputeAccess](#) 和 [GetInstanceAccess](#) 的文件，以根據使用中的 Amazon GameLift Servers 伺服器 SDK 版本來釐清遠端存取程序。
 - 修訂 [GameSessionPlacement](#) 的說明，以記錄放置處於「待定」狀態時，遊戲工作階段資訊如何暫時存在。

2023 年 7 月 13 日：Amazon GameLift Servers 新增機群硬體指標

您現在可以追蹤 Amazon GameLift Servers 受管 EC2 機群的硬體效能指標。指標包括 CPU 使用率、網路流量磁碟區和磁碟讀取/寫入活動的 EC2 執行個體指標。對於 Amazon GameLift Servers，這些指標會描述機群位置中的所有作用中執行個體。您可以使用 中的 Amazon CloudWatch 儀表板來檢視這些

機群硬體指標 AWS Management Console。您也可以在 Amazon GameLift Servers 主控台的機群詳細資訊中檢視它們。

進一步了解：

- [使用 Amazon CloudWatch 監控 Amazon GameLift Servers](#) (機群的指標)、Amazon GameLift Servers 開發人員指南

2023 年 6 月 29 日：Amazon GameLift Servers 啟動對 Amazon Linux 2023 的支援

更新的 SDK 版本：AWS SDK 1.11.111

Amazon GameLift Servers 客戶現在可以使用 Amazon Linux 2023 作業系統來託管其遊戲伺服器。AL2023 提供數個優於 AL2 的改進，包括安全性。此作業系統適用於所有，AWS 區域 但中國區域除外。

客戶可以使用較新的 Linux 作業系統，並在 2023 年 12 月 Amazon Linux (AL1) 支援結束時繼續接收重要的安全性更新。Amazon Linux 2 的支援會持續到 2025 年 6 月 30 日。

進一步了解：

- [Amazon GameLift Servers Linux 伺服器常見問答集](#)
- [比較 Amazon Linux 2 和 Amazon Linux 2023](#)
- Amazon GameLift Servers API 參考連結：
 - [AWS SDK 動作 CreateBuild](#)
 - [CLI 命令 upload-build](#)
 - [CLI 命令 create-build](#)

2023 年 5 月 25 日：Amazon GameLift Servers FleetIQ 新增篩選條件以排除耗盡執行個體上的遊戲工作階段置放

已更新 SDK 版本：AWS SDK 1.11.87

如果您使用 Amazon GameLift Servers FleetIQ 進行遊戲託管，您現在可以防止遊戲工作階段放置在目前正在耗盡的執行個體上。耗盡的執行個體會標記為關閉，但如果沒有其他託管資源可用，仍然可以選取它們來託管新的遊戲工作階段。透過這項新功能，您可以排除完全耗盡執行個體的使用。

呼叫時請使用此功能 `ClaimGameServer` 來尋找可用的遊戲伺服器。新增新 `FilterOption` 參數，並將允許的執行個體狀態設定為僅限 `ACTIVE`。在回應中，只會在搜尋和申請可用的遊戲伺服器時 Amazon GameLift Servers FleetIQ 查看作用中的執行個體。

進一步了解：

- Amazon GameLift Servers API 參考中的 [ClaimGameServer](#)
- 《Amazon GameLift Servers FleetIQ 開發人員指南》中的 [如何 FleetIQ 運作](#)

2023 年 5 月 16 日：Amazon GameLift Servers 支援機群的成本分配標記

Amazon GameLift Servers 客戶現在可以使用 AWS Billing 成本分配標籤來組織遊戲託管成本。您可以將成本分配標籤指派給個別 Amazon GameLift Servers EC2 機群資源，以追蹤機群如何為整體託管成本做出貢獻。

進一步了解：

- [管理您的 Amazon GameLift Servers 託管成本](#)
- [使用 AWS 成本分配標籤](#)、AWS Billing 使用者指南

2023 年 4 月 20 日：Amazon GameLift Servers 啟動對 Windows Server 2016 的支援

更新的 SDK 版本：AWS SDK 1.11.63

Amazon GameLift Servers 客戶現在可以使用 Windows Server 2016 作業系統來託管遊戲伺服器。此作業系統適用於所有 AWS 區域。當 Microsoft 在 2023 年 10 月終止對 Windows Server 2012 的支援時，客戶可以使用較新的 Windows 作業系統並繼續接收重要的安全更新。

從今天開始，需要 Windows 執行期環境的新客戶在建立新的遊戲伺服器組建以進行託管時，必須指定 Windows Server 2016。現有客戶可以繼續使用 Windows Server 2012 建立新的組建和機群，但必須在 2023 年 10 月 10 日 Microsoft 支援結束日期之前，使用 Windows Server 2016 完成遷移。

此更新包含下列服務變更：

- 使用 Amazon GameLift Servers SDK 或 CLI 命令建立遊戲伺服器組建時，您現在必須明確設定作業系統。不再有預設值。若要在 Windows Server 2016 上部署遊戲伺服器，請使用值 `WINDOWS_2016`。

- 使用Amazon GameLift Servers主控台建立遊戲伺服器組建時，您必須從可用的值中選取作業系統。如果您是具有作用中 Windows Server 2012 機群的現有客戶，您可以選擇 WINDOWS_2012或 WINDOWS_2016。

進一步了解：

- Amazon GameLift Servers API 參考連結：
 - [CLI 命令 upload-build](#)
 - [CLI 命令 create-build](#)
 - [AWS SDK 動作 CreateBuild](#)
- [Amazon GameLift Servers Windows 2012 常見問答集](#)

2023 年 4 月 13 日：Amazon GameLift Servers啟動適用於 Unreal 的伺服器 SDK 5.x

已更新 SDK 版本：適用於 Unreal 的伺服器 SDK 5.0.0

Unreal Engine 的Amazon GameLift Servers最新版本輕量型外掛程式現在是以Amazon GameLift Servers伺服器 SDK 5.x 為基礎。若要開始整合 Unreal Engine 環境與，Amazon GameLift Servers請參閱下列連結。

進一步了解：

- [Amazon GameLift Servers 整合至 Unreal Engine 專案](#)
- [Amazon GameLift Servers 新增至您的遊戲伺服器](#)
- [適用於 -- 動作的 C++ 伺服器 SDK Amazon GameLift Servers 5.x](#)

2023 年 3 月 14 日：Amazon GameLift Servers 推出新的主控台體驗

新的Amazon GameLift Servers主控台包含下列改進項目：

- 改善導覽 – 新的導覽窗格有助於在Amazon GameLift Servers資源之間進行導覽。
- Amazon GameLift Servers 登陸頁面 – 新的登陸頁面提供實用文件的連結、顯示的高階概觀Amazon GameLift Servers，以及透過文件連結、常見問答集和提供支援 AWS re:Post。
- 改善的 Amazon CloudWatch 指標 – Amazon GameLift Servers指標現在可在Amazon GameLift Servers主控台和 CloudWatch 儀表板中使用。此更新也包含效能、使用率和玩家工作階段的新指標。

進一步了解：

- [在Amazon GameLift Servers主控台中追蹤遊戲託管](#)
- [建置FlexMatch配對建構器](#)

2023 年 2 月 14 日：Amazon GameLift Servers 現在支援 Amazon SNS 主題的伺服器端加密

SNS 主題的伺服器端加密 ((SSE)) 會加密您的靜態敏感資料。SSE 使用 AWS Key Management Service (AWS KMS) 金鑰來保護 SNS 主題的內容。

進一步了解：

- [設定遊戲工作階段放置的事件通知](#)
- [FlexMatch 配對事件](#)
- [靜態加密](#)

2023 年 2 月 9 日：Amazon GameLift Servers伺服器 SDK 支援 .NET 6 搭配 C#10

已更新 SDK 版本：適用於 .NET 6 的伺服器 SDK 5.0.0。不需要任何 SDK 更新。

如果您使用 Unity 即時開發平台，請繼續搭配 .NET 4.6 使用Amazon GameLift Servers伺服器 SDK 5.0.0。Unity 不支援 .NET 6。

進一步了解：

- [Amazon GameLift Servers 開始使用](#)時下載最新版本的Amazon GameLift Servers伺服器 SDK
- [適用於 -- 動作的 C# 伺服器 SDK Amazon GameLift Servers 5.x](#)

2023 年 1 月 31 日：Amazon GameLift Servers伺服器 SDK 支援 Go 語言

已更新 SDK 版本：適用於 Go 的伺服器 SDK 5.0.0

進一步了解：

- [Amazon GameLift Servers 開始使用](#)時下載最新版本的Amazon GameLift Servers伺服器 SDK
- [適用於 Amazon GameLift Servers -- 動作的 Go 伺服器 SDK](#)

2022 年 12 月 1 日：Amazon GameLift Servers 啟動 Amazon GameLift ServersAnywhere 和 Amazon GameLift Servers Server SDK 5.0

已更新 SDK 版本：AWS SDK 1.10.21、適用於 C++ 和 C# 的伺服器 SDK 5.0.0

Amazon GameLift ServersAnywhere 使用您的遊戲伺服器資源來託管 Amazon GameLift Servers 遊戲伺服器。您可以使用 Amazon GameLift ServersAnywhere 將自己的運算資源與 Amazon GameLift Servers 受管 EC2 運算整合，將遊戲伺服器分散到多種運算類型。您也可以使用 Amazon GameLift ServersAnywhere 反覆測試遊戲伺服器，而無需 Amazon GameLift Servers 針對每次反覆上傳建置到。

重點：

- 新的 Amazon GameLift ServersAnywhere 機群和運算類型
- Amazon GameLift ServersAnywhere 運算資源註冊
- 改善測試反覆運算週期

Amazon GameLift Servers 伺服器 SDK 5.0.0 為現有的伺服器 SDK 和新的資源類型運算引進了改進。伺服器 SDK 5.0.0 支援 Amazon GameLift ServersAnywhere 和使用您自己的運算資源進行遊戲伺服器託管。

進一步了解：

- [適用於 的伺服器 SDK 5.x Amazon GameLift Servers](#)
- [機群位置](#)
- [選擇受管機群的運算資源](#)
- [建立 Amazon GameLift ServersAnywhere 機群](#)

2022 年 8 月 25 日：Amazon GameLift Servers 啟動對 Local Zones 的支援

更新的 SDK 版本：AWS SDK 1.9.333

Amazon GameLift Servers 現已在美國的八個本地區域提供，因此您可以部署更接近玩家的機群。您可以將 Local Zones 新增至機群，以將所有受管 Amazon GameLift Servers 功能與 Local Zones 搭配使用。

Local Zones 將 AWS 資源和服務擴展到雲端邊緣，接近大型人口、產業和資訊技術 (IT) 中心。這表示您可以部署需要單一位數毫秒延遲的應用程式，使其更接近最終使用者或內部部署資料中心。

進一步了解：

- [Amazon GameLift Servers 本機區域](#)
- [機群位置](#)
- [建立Amazon GameLift Servers受管 EC2 機群](#)

2022 年 6 月 28 日：Amazon GameLift Servers 推出新的選擇加入主控台體驗

新的Amazon GameLift Servers主控台包含下列改進項目：

- 改善導覽 – 新的導覽窗格有助於在Amazon GameLift Servers資源之間進行導覽。
- Amazon GameLift Servers 登陸頁面 – 新的登陸頁面提供實用文件的連結、顯示的高階概觀Amazon GameLift Servers，以及透過文件連結、常見問答集和提供支援 AWS re:Post。
- 改善的 Amazon CloudWatch 指標 – Amazon GameLift Servers指標現在可在Amazon GameLift Servers主控台和 CloudWatch 儀表板中使用。此更新也包含效能、使用率和玩家工作階段的新指標。

進一步了解：

- [在Amazon GameLift Servers主控台中追蹤遊戲託管](#)
- [建置FlexMatch配對建構器](#)

2022 年 2 月 15 日：FlexMatch新增複合規則和其他改進

FlexMatch 使用者現在可以存取下列功能：

- 複合規則 – 新增對 40 名或更少玩家配對的複合配對規則的支援。您現在可以使用邏輯陳述式來建立複合規則來形成相符項目。如果規則集中沒有複合規則，則規則集中的所有規則都必須為 true，才能形成相符項目。透過複合規則，您可以使用下列邏輯運算子來選擇要套用的規則：and、not、or和 xor。
- 彈性的團隊選擇 – 更新配對屬性表達式，以支援選取所有可用團隊的子集。
- 較長的字串清單 – 將玩家屬性值的字串清單中的字串數目上限從 10 個增加到 100 個。

進一步了解：

- [Amazon GameLift ServersFlexMatch 開發人員指南](#)：

- [FlexMatch 規則類型](#)
- [FlexMatch 屬性表達式](#)
- [AttributeValue : SL](#)

2021 年 10 月 28 日：Amazon GameLift Servers 新增對亞太區域（大阪）區域中多區域機群的支援；Amazon GameLift ServersFleetIQ 新增對 AWS Graviton2 處理器的支援

更新的 SDK 版本：AWS SDK [1.9.133](#)

亞太區域 (大阪) 區域現可使用 Amazon GameLift Servers。遊戲開發人員現在可以使用 GameLift 多區域機群在大阪部署執行個體。

您現在可以根據 Arm 型處理器架構使用 Graviton2-hosted遊戲伺服器，相較於同等的 Intel 型運算選項，以較低的成本實現更高的效能。

重點：

- 亞太區域 (大阪) 區域現可使用 Amazon GameLift Servers。
- Amazon GameLift Servers FleetIQ 遊戲伺服器群組現在可以設定為管理 Graviton2 執行個體系列 c6g、m6g 和 r6g。

進一步了解：

- [Amazon GameLift Servers 多區域機群](#)
- [CreateGameServerGroup](#)
- [AWS graviton 處理器](#)

2021 年 9 月 20 日：Amazon GameLift Servers發行 Unity 的外掛程式

Unity 1.0.0 版的Amazon GameLift Servers外掛程式包含程式庫和原生 UI，可讓您更輕鬆地存取 Amazon GameLift Servers資源並整合Amazon GameLift Servers到您的 Unity 遊戲。您可以使用 Unity 的 Amazon GameLift Servers外掛程式來存取 Amazon GameLift Servers APIs，並針對常見的遊戲案例部署 AWS CloudFormation 範本。外掛程式也包含適用於範例案例的範例遊戲。您可以使用 Amazon GameLift Servers Local 來查看遊戲用戶端和遊戲伺服器之間傳遞的訊息，以了解典型的遊戲如何與 互動Amazon GameLift Servers。

Unity 的外掛程式支援 Unity 2019.4 LTS 和 2020.3 LTS。

重點：

- 使用不同的案例建置、執行和修改範例遊戲，或建立您自己的遊戲。
- 部署一般遊戲案例的範例 AWS CloudFormation 案例，包括僅限身分驗證、單一區域機群、具有佇列和自訂配對建構器的多區域機群、具有佇列和自訂配對建構器的 Spot 機群，以及 FlexMatch。

進一步了解：

- [將遊戲與 Unity 的 Amazon GameLift Servers 外掛程式整合](#)

2021 年 6 月 30 日：FlexMatch 新增 batchDistance 規則

您可以使用 batchDistance 規則類型來指定字串或數值屬性，為每個區段帶來許多好處。

重點：

- 對於大型配對 (>40 名玩家)，您現在可以根據技能、模式和地圖取得相同的平衡，而不是僅根據技能平均平衡玩家。確保配對中的每個人都位於技能範圍中、帶定多個數值屬性，例如聯盟或遊戲樣式，以及根據地圖或遊戲模式等字串屬性分組。您也可以隨著時間建立擴展。例如，您可以建立擴展，以允許更大的技能水準範圍輸入玩家等待的時間越長。

對於 40 名玩家以下的配對，您可以使用新的簡化規則表達式。

2021 年 6 月 3 日：Amazon GameLift Servers 即時用戶端 SDK 和伺服器 SDK 更新

更新 SDK 版本：適用於 Unreal 的即時用戶端 SDK 1.2.0、伺服器 SDK 3.4.0

透過此最新的 SDK 更新，您現在可以將 IL2CPP 整合到使用 RTS 用戶端 SDK 的行動應用程式中，並遵循架構的最佳實務。您現在可以建置適用於 Unreal 的 Amazon GameLift Servers 伺服器開發套件 4.26 版。此更新包含與您的 Windows 或 Linux 遊戲伺服器整合的元件，包括 Amazon GameLift Servers 伺服器 SDK、Amazon GameLift ServersLocal 和 Unreal Engine 外掛程式的 C++ 和 C# 版本。

重點：

- 新增 RTS 用戶端 SDK 中對 IL2CPP 的支援，以及將原生程式庫建置為架構，因此您可以為最新的行動裝置建置 RTS 用戶端。

- 您可以使用 [DescribePlayerSessions\(\)](#) 取得單一玩家工作階段、遊戲工作階段中所有玩家工作階段，或與單一玩家 ID 相關聯的所有玩家工作階段的資訊。
- 您可以使用 [GetInstanceCertificate\(\)](#) 來擷取與機群及其執行個體相關聯的 PEM 編碼 TLS 憑證的檔案位置。
- Unreal 4.26 版的已建立伺服器 SDK 支援。
- 現有的 C# SDK 4.0.2 版已驗證與 Unity 2020.3 相容。不需要任何 SDK 更新。

進一步了解：

- [Amazon GameLift Servers 開發人員指南](#)：
 - [DescribePlayerSessions\(\)](#)
 - [GetInstanceCertificate\(\)](#)

2021 年 3 月 23 日：將通知 Amazon GameLift Servers 新增至遊戲工作階段置放

更新的 SDK 版本：AWS SDK [1.8.168](#)

您現在可以使用事件來監控遊戲工作階段佇列的遊戲工作階段置放活動。建立 Amazon Simple Notification Service (Amazon SNS) 主題以發佈事件通知，或使用 CloudWatch Events 設定事件追蹤。

重點：

- 對於每個佇列，您可以設定要包含在所有事件傳訊中的自訂文字字串。
- 使用 Amazon SNS 主題時，您可以設定其他存取條件，限制發佈至特定佇列。

進一步了解：

- Amazon GameLift Servers 開發人員指南：
 - [設定遊戲工作階段放置的事件通知](#) (新)
 - [遊戲工作階段置放事件](#) (新)
- [API 參考 \(AWS SDK\)](#)
 - 新的遊戲工作階段佇列參數 NotificationTarget 和 CustomEventData：[GameSessionQueue](#)、[CreateGameSessionQueue](#)、[UpdateGameSessionQueue](#)
- [Amazon GameLift Servers 論壇](#)

2021 年 3 月 16 日：Amazon GameLift Servers 新增多區域機群、六個新區域

更新的 SDK 版本：AWS SDK [1.8.163](#)

Amazon GameLift Servers 受管託管現在可在 21 AWS 區域使用。新區域包括開普敦 (af-south-1)、巴林 (me-south-1)、香港 (ap-east-1)、米蘭 (eu-south-1)、巴黎 (eu-west-3) 和斯德哥爾摩 (eu-north-1)。

使用新的 Amazon GameLift Servers 多位置機群功能，您現在可以設定單一機群，在 20 個 Amazon GameLift Servers 支援的任何或所有區域中託管遊戲伺服器（北京區域除外）。此功能旨在大幅減少在全球設定和維護 Amazon GameLift Servers 託管資源所需的工作。您可以在下列 AWS 區域中建立多位置機群：us-east-1（維吉尼亞北部）、us-west-2（奧勒岡）、eu-central-1（法蘭克福）、eu-west-1（愛爾蘭）、ap-southeast-2（雪梨）、ap-northeast-1（東京）和 ap-northeast-2（首爾）。在所有其他區域中，您可以視需要繼續設定單一位置機群。在此版本之前建立的所有機群都是單一位置機群。使用多位置機群不會影響您的託管成本。Amazon GameLift Servers 定價取決於您使用的執行個體類型、位置和數量。（如需詳細資訊，請參閱 [Amazon GameLift Servers 定價](#)。）多位置機群的 AWS CloudFormation 支援即將推出。

Note

多位置機群不適用於中國區域。位於中國區域的 Amazon GameLift Servers 資源無法與其他 Amazon GameLift Servers 區域中的資源互動或使用。

重點：

- 使用多位置機群，明確新增遠端位置清單。Amazon GameLift Servers 會將相同類型和組態的執行個體，包括建置和執行時間組態，部署到機群的主區域和所有新增的位置。
- 獨立調整每個位置的容量設定和擴展。自動擴展政策適用於整個機群，但您可以依位置開啟或關閉它們。
- 在特定機群位置啟動新的遊戲工作階段。使用遊戲工作階段佇列或配對來放置遊戲工作階段時，您現在可以優先考慮新遊戲工作階段從位置開始的位置、託管成本和玩家延遲。
- 在 Amazon GameLift Servers 主控台中取得託管指標，針對機群中的所有位置彙總，或依每個機群位置細分。

進一步了解：

- [Amazon 遊戲技術部落格](#)

- [API 參考 \(AWS SDK\)](#)

- 新的機群位置操

- 作：[CreateFleetLocations](#)、[DescribeFleetLocationAttributes](#)、[DescribeFleetLocationCapacity](#)、[DescribeFleetLocations](#)

- 已更新機群操作，並支援新的多位

- 置：[CreateFleet](#)、[UpdateFleetCapacity](#)、[DescribeEC2InstanceLimits](#)、[DescribeInstances](#)、[StopFleetA](#)

- 已更新遊戲工作階段置放操作，具有新的優先順序和篩選功

- 能：[CreateGameSessionQueue](#)、[DescribeGameSessionQueues](#)、[UpdateGameSessionQueues](#)

- 已更新遊戲工作階段建立操作，並支援新的位

- 置：[CreateGameSession](#)、[DescribeGameSessions](#)、[DescribeGameSessionDetails](#)、[SearchGameSes](#)

- [Amazon GameLift Servers 開發人員指南](#)：

- [Amazon GameLift Servers 服務位置](#)（已更新）

- [自訂您的 Amazon GameLift Servers EC2 受管機群](#) (新)

- [使用 擴展遊戲託管容量 Amazon GameLift Servers](#)（已更新）

- [自訂遊戲工作階段佇列](#) (新)

- [Amazon GameLift Servers 主控台](#)中的機群詳細資訊（已更新）

- [Amazon GameLift Servers 論壇](#)

2021 年 2 月 9 日：Amazon GameLift Servers 擴展獨立 AMD 執行個體的支援 FlexMatch

更新的 SDK 版本：AWS SDK [1.8.139](#)

此版本包含下列更新：

- Amazon GameLift Servers FleetIQ 遊戲伺服器群組現在可以設定為管理 AMD 執行個體系列 C5a, M5a 和 R5a。如 GameServerGroup [InstanceDefinition](#) 所列，支援的 Amazon EC2 執行個體類型現在包含下列項目：

- c5a.large、c5a.xlarge、c5a.2xlarge、c5a.4xlarge、c5a.8xlarge、c5a.12xlarge、c5a.16xlarge、c5a.24xlarge
 - m5a.large、m5a.xlarge、m5a.2xlarge、m5a.4xlarge、m5a.8xlarge、m5a.12xlarge、m5a.16xlarge、m5a.24xlarge
 - r5a.large、r5a.xlarge、r5a.2xlarge、r5a.4xlarge、r5a.8xlarge、r5a.12xlarge、r5a.16xlarge、r5a.24xlarge

注意：的 AMD 執行個體FleetIQ目前無法在中國（北京）AWS 區域使用。請參閱中國的功能[可用性和實作差異](#)。

- Amazon GameLift Servers 受管遊戲託管現在支援中國（北京）區域的 AMD 執行個體，由 Sinnet 營運。新的 AMD 執行個體系列包括 M5a 和 R5a。機群 [InstanceType](#) 列出的支援 EC2 執行個體類型現在包含下列項目：
 - m5a.large、m5a.xlarge、m5a.2xlarge、m5a.4xlarge、m5a.8xlarge、m5a.12xlarge、m5a.16xlarge、m5a.24xlarge
 - r5a.large、r5a.xlarge、r5a.2xlarge、r5a.4xlarge、r5a.8xlarge、r5a.12xlarge、r5a.16xlarge、r5a.24xlarge
- Amazon GameLift Servers FlexMatch 現在可以在中國（北京）區域做為獨立的配對解決方案，由 Sinnet 營運。客戶可以在北京區域建立 FlexMatch 配對建構器，並將 [FlexMatchMode](#) 參數設定為 STANDALONE。如需使用 FlexMatch Amazon GameLift Servers 受管託管或非 Amazon GameLift Servers 託管解決方案之的詳細資訊，請參閱 [Amazon GameLift Servers FlexMatch 開發人員指南](#)。
- 設定 Amazon GameLift Servers 的事件通知時 FlexMatch，您現在可以指定 Amazon SNS FIFO 主題做為通知目標。如需詳細資訊，請參閱：
 - [MatchmakingConfiguration NotificationTarget](#)，Amazon GameLift Servers API 參考
 - [設定 FlexMatch 事件通知](#)，Amazon GameLift Servers FlexMatch 開發人員指南
 - [Amazon SNS FIFO 簡介 – First-in-first-out 的 Pub/Sub 訊息](#)、AWS 新聞部落格

2020 年 12 月 22 日：Amazon GameLift Servers 伺服器 SDK 支援 Unreal Engine 4.25 和 Unity 2020

已更新 SDK 版本：Amazon GameLift Servers 伺服器 SDK 4.0.2、Unreal 外掛程式 3.3.3 版

Amazon GameLift Servers Server SDK 的最新版本包含下列元件：

- 更新的 Unreal 外掛程式已更新，可與 Unreal Engine 4.25 相容。API 未變更。
- 現有的 C# SDK 4.0.2 版已驗證與 Unity 2020 相容。不需要任何 SDK 更新。

[Amazon GameLift Servers 開始使用](#) 時下載最新版本的 Amazon GameLift Servers 伺服器開發套件。

2020 年 11 月 24 日：Amazon GameLift Servers FlexMatch 現在適用於隨處託管的遊戲

已更新 SDK 版本：AWS SDK [1.8.95](#)

Amazon GameLift Servers FlexMatch 是多玩家遊戲的可自訂配對服務。最初專為 Amazon GameLift Servers 受管託管的使用者設計，FlexMatch 現在可以整合到使用其他託管系統的遊戲中，包括 peer-to-peer、專屬內部部署運算和雲端運算基本類型。用於 Amazon EC2 上遊戲託管 Amazon GameLift Servers FleetIQ 的遊戲現在可以使用 實作配對 FlexMatch。

FlexMatch 提供強大的配對演算法和規則語言，為您提供寬廣的緯度來自訂配對程序，以便根據關鍵玩家特性和報告的延遲來配對玩家。此外，FlexMatch提供配對請求工作流程，支援玩家隊伍、玩家接受度和配對回填等功能。當您FlexMatch搭配 Amazon GameLift Servers 受管託管使用 或 時Amazon GameLift ServersRealtime，配對建構器會自動使用 Amazon GameLift Servers 尋找託管資源，並為新形成的配對啟動新的遊戲工作階段。使用 FlexMatch 做為獨立服務時，配對建構器會將配對結果傳回您的遊戲，然後可以使用您的託管解決方案啟動新的遊戲工作階段。

的 API 操作FlexMatch是 Amazon GameLift Servers服務 API 的一部分，包含在 AWS SDK 和 AWS Command Line Interface () 中AWS CLI。此版本包含這些更新以支援獨立配對：

- API 資源MatchmakingConfiguration有下列變更：
 - 新屬性，FlexMatchMode指出配對建構器是否與Amazon GameLift Servers受管託管搭配使用，或作為獨立配對。
 - FlexMatchMode 當 設為獨立時，GameSessionQueueArns不需要 屬性。
 - 這些屬性不會與獨立配對搭配使用：AdditionalPlayerCount、BackfillMode、GameProperties、GameSessionData。
- 自動回填功能不適用於獨立配對。

2020 年 11 月 24 日：AMD 執行個體現已於 推出 Amazon GameLift Servers

更新的 SDK 版本：AWS SDK [1.8.95](#)

現在支援的 Amazon EC2 執行個體類型清單Amazon GameLift Servers包含三個新的執行個體系列：C5a, M5a 和 R5a。這些系列由以高達 3.3 的頻率執行的 AMD EPYC 處理器提供支援的 AMD 運算最佳化執行個體組成。GHz. AMD 執行個體與 x86 相容；目前在上執行的遊戲Amazon GameLift Servers可以部署到 AMD 執行個體類型，無需變更。新的執行個體可在下列 AWS 區域使用：美國東部（維吉尼亞北部和俄亥俄）、美國西部（奧勒岡北部和加利佛尼亞北部）、加拿大中部（蒙特婁）、南美洲（聖保羅）、歐洲中部（法蘭克福）、歐洲西部（倫敦和愛爾蘭）、亞太區域南部（孟買）、亞太區域東北部（首爾和東京）和亞太區域東南部（新加坡和雪梨）。

新的 AMD 執行個體包括：

- c5a.large、c5a.xlarge、c5a.2xlarge、c5a.4xlarge、c5a.8xlarge、c5a.12xlarge、c5a.16xlarge、c5a.24xlarge
- m5a.large、m5a.xlarge、m5a.2xlarge、m5a.4xlarge、m5a.8xlarge、m5a.12xlarge、m5a.16xlarge、m5a.24xlarge
- r5a.large、r5a.xlarge、r5a.2xlarge、r5a.4xlarge、r5a.8xlarge、r5a.12xlarge、r5a.16xlarge、r5a.24xlarge

進一步了解：

- [Amazon 遊戲技術部落格](#)
- [Amazon GameLift Servers 執行個體定價](#)
- [採用 AMD EPYC 處理器的 Amazon EC2 執行個體](#)
- [Amazon GameLift Servers 論壇](#)

2020 年 11 月 11 日：Amazon GameLift Servers 伺服器 SDK 版本更新

已更新 SDK 版本：Amazon GameLift Servers 伺服器 SDK 4.0.2

新的伺服器開發套件 4.0.2 版修正了 API 操作的已知問題。StartMatchBackfill() 此操作現在會傳回相符回填請求的正確回應。

此問題不會影響配對回填程序，而且此功能的運作方式沒有變更。此問題可能會影響相符回填請求的日誌訊息和錯誤處理。

[Amazon GameLift Servers 開始使用](#)時下載最新版本的 Amazon GameLift Servers 伺服器開發套件。

2020 年 11 月 5 日：新的 FlexMatch 演算法自訂

FlexMatch 使用者現在可以針對配對程序調整下列預設行為。這些自訂會在配對規則集中設定。Amazon GameLift Servers SDKs 沒有變更。

- 排定回填票證的優先順序：您可以選擇在搜尋可接受的相符項目時，提高或降低配對回填票證的優先順序。啟用自動回填功能時，排定回填票證的優先順序非常有用。使用演算法屬性 backfillPriority。
- 預先排序以最佳化配對一致性和效率：設定配對建構器在批次處理票證進行評估之前預先排序票證集區。透過根據關鍵玩家屬性預先排序門票，您產生的配對往往會有在這些屬性中更相似的玩家。您也可以在此對規則中使用的相同屬性上預先排序，以提升評估程序的效率。使用演算法屬性 sortByAttributes，並將 strategy 屬性設定為「排序」。
- 調整擴展等待時間的觸發方式：根據最新（預設）或未完成配對中最舊票證的存留期，在觸發擴展之間進行選擇。在最舊的票證上觸發往往會更快地完成配對，而在最新的票證上觸發會導致更高的配對品質。使用演算法屬性 expansionAgeSelection。

2020 年 9 月 17 日：Amazon GameLift Servers 更新伺服器 SDK

更新的 SDK 版本：Amazon GameLift Servers 伺服器 SDK 4.0.1

新的伺服器開發套件包含下列更新：

- C# API 4.0.1 版
 - [TerminateGameSession\(\)](#) 不再支援 API 操作。以呼叫 取代 [ProcessEnding\(\)](#) 以結束遊戲工作階段和伺服器程序。
 - [GetInstanceCertificate\(\)](#) 已修正 操作的已知問題。
 - 操作[GetTerminationTime\(\)](#)現在會傳回資料類型 `AwsDateTimeOutcome` 的值。
- C++ API 3.4.1 版
 - [TerminateGameSession\(\)](#) 不再支援 操作。將其取代為呼叫 [ProcessEnding\(\)](#)以結束遊戲工作階段和伺服器程序。
- Unreal Engine 外掛程式 3.3.2 版
 - [TerminateGameSession\(\)](#) 不再支援 操作。將其取代為呼叫 [ProcessEnding\(\)](#)以結束遊戲工作階段和伺服器程序。
 - 回呼操作OnUpdateGameSession會新增至 [FProcessParameters](#) , 以支援配對回填。

[Amazon GameLift Servers 開始使用](#)時下載最新版本的Amazon GameLift Servers伺服器開發套件。

2020 年 8 月 27 日：Amazon GameLift ServersFleetIQ適用於使用 Amazon EC2 的遊戲託管（一般可用性）

已更新 SDK 版本：AWS SDK [1.8.36](#)

Amazon EC2 上低成本、雲端型遊戲託管Amazon GameLift ServersFleetIQ的解決方案現在已全面推出。Amazon GameLift Servers FleetIQ 可讓開發人員透過最佳化遊戲託管的可行性，直接在 Amazon EC2 Spot 執行個體上託管遊戲伺服器。遊戲開發人員可以使用 Amazon GameLift Servers FleetIQ 搭配新遊戲或補充現有遊戲的容量。此解決方案支援使用容器或其他 AWS 服務，例如 AWS Shield 和 Amazon Elastic Container Service (Amazon ECS)。

此一般可用性版本包含解決方案的下列更新Amazon GameLift ServersFleetIQ：

- 新的 API 操作會在Amazon GameLift ServersFleetIQ遊戲伺服器群組的所有作用中執行個體上DescribeGameServerInstances傳回資訊，包括狀態。
- 新的平衡策略 會將遊戲伺服器群組ON_DEMAND_ONLY設定為僅使用隨需執行個體。您可以隨時更新遊戲伺服器群組的平衡策略，以便視需要在 Spot 執行個體和隨需執行個體之間切換。
- 下列預覽元素已因一般可用性而遭到捨棄：
 - 使用遊戲伺服器資源的自訂排序索引鍵。遊戲伺服器可根據註冊時間戳記進行排序。
 - 遊戲伺服器資源的標記。

2020 年 4 月 16 日：Amazon GameLift Servers 更新 Unity 和 Unreal Engine 的伺服器 SDK

更新 SDK 版本：Amazon GameLift Servers 伺服器 SDK 4.0.0、Amazon GameLift Servers 本機 1.0.5

最新版本的 Amazon GameLift Servers 伺服器開發套件包含下列更新元件：

- 已更新 Unity 2019 的 C# SDK 4.0.0 版。
- 針對 Unreal Engine 4.22、4.23 和 4.24 版更新了 Unreal 外掛程式 3.3.1 版。
- Amazon GameLift Servers 本機 1.0.5 版已更新，以測試使用 C# 伺服器 SDK 4.0.0 版的整合。

[Amazon GameLift Servers 開始使用](#)時下載最新版本的 Amazon GameLift Servers 伺服器開發套件。

2020 年 4 月 2 日：Amazon GameLift Servers FleetIQ 適用於 EC2 上的遊戲託管（公開預覽版）

更新的 SDK 版本：AWS SDK [1.7.310](#)

Amazon GameLift Servers FleetIQ 功能可將低成本 Spot 執行個體的可行性最佳化，以進行遊戲託管。此功能的適用範圍現在已擴展至想直接管理託管資源，而不是透過受管 Amazon GameLift Servers 服務進行管理的客戶。此解決方案支援使用容器或其他 AWS 服務，例如 AWS Shield 和 Amazon Elastic Container Service (Amazon ECS)。

進一步了解：

上的 [GameTech 部落格文章](#) Amazon GameLift Servers FleetIQ

2019 年 12 月 19 日：改善 AWS 資源的資源管理 Amazon GameLift Servers

更新的 SDK 版本：AWS SDK [1.7.249](#)

您現在可以利用 AWS 資源管理工具 Amazon GameLift Servers。特別是，所有關鍵 Amazon GameLift Servers 資源 - 組建、指令碼、機群、遊戲工作階段佇列、配對組態和配對規則集 - 現在都會指派 Amazon Resource Name (ARN) 值。資源 ARN 提供在所有 AWS 區域中唯一的一致識別符。它們可用於建立資源特定 AWS Identity and Access Management (IAM) 許可政策。資源現在會獲指派 ARN 和預先存在的資源識別符，這並非區域特定。

此外，Amazon GameLift Servers 資源現可支援標記。您可以使用標籤來組織資源、建立 IAM 許可政策來管理對資源群組的存取、自訂 AWS 成本明細等。為 Amazon GameLift Servers 資源管理

標籤時，請使用 Amazon GameLift Servers API 動作 `TagResource()`、`UntagResource()` 和 `ListTagsForResource()`。

進一步了解：

- Amazon GameLift Servers API 參考中的 [TagResource](#)
- AWS 一般參考中的 [標記 AWS 資源](#)
- AWS 一般參考中的 [Amazon 資源名稱](#)

2019 年 11 月 14 日：中國（北京）區域的新 AWS CloudFormation 範本、更新

更新的 SDK 版本：AWS SDK [1.7.210](#)

AWS CloudFormation 的範本 Amazon GameLift Servers

Amazon GameLift Servers 現在可以透過 建立和管理 資源 AWS CloudFormation。現有的 AWS CloudFormation 建置和機群範本已更新以符合目前的資源，而且新的範本現在可用於指令碼、佇列、配對組態和配對規則集。AWS CloudFormation templates 可大幅簡化管理相關 AWS 資源群組的任務，特別是在跨多個區域部署遊戲時。

進一步了解：

- AWS CloudFormation 《使用者指南》中的 [Amazon GameLift Servers 資源類型參考](#)
- [使用 管理 Amazon GameLift Servers 託管資源 AWS CloudFormation](#) 《Amazon GameLift Servers 開發人員指南》中的

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。